Big Data for Mobility Tracking Knowledge Extraction in Urban Areas

D4.1 Analytics for Mobility Patterns Detection and Forecasting

Document Summary Information

Grant Agreement No	780754	Acronym	TRACK&KNOW
Full title	Big Data for Mobility Tracking Knowledge Extraction in Urban Areas		
Start Date	01/01/2018	Duration	36 months
Project URL	https://trackandknow.eu/		
Deliverable	D4.1 Analytics for mobility patterns detection and forecasting		
Work Package	WP4 Big Data Analytics Toolboxes (BDA Toolbox)		
Contractual due date	30/6/2020	Actual submission date	30/6/2020
Nature	Report	Dissemination Level	PU
Lead Beneficiary	UPRC		
Contributions From	Eva Chondrodima (UPRC), Christos Doulkeridis (UPRC), Harris Georgiou (UPRC), Panagiotis Nikitopoulos (UPRC), Panagiotis Tampakis (UPRC) Mirco Nanni (CNR), Agnese Bonavita (CNR), Roberto Trasarti (CNR) Feng Liu (UHASSELT), Cheng Fu (UZH), Ibad Kureshi (ILS)		
Responsible Author	Harris Georgiou (UPRC)		

HISTORY OF CHANGES

Version	Date	Changes	Author
0.1	7/2/2020	Initialization / Table of Contents	H. Georgiou
0.2	20/5/2020	Conversion to LaTeXformat	H. Georgiou
0.3a	5/6/2020	Section 3.5 contributions (UPRC)	E. Chondrodima
0.3b	6/6/2020	Section 3.8 contributions (UHasselt)	F. Liu
0.3c	7/6/2020	Section 3.2-3.3 contributions (CNR)	M. Nanni, A. Bonavita, R. Trasarti
0.3d	7/6/2020	Section 3.9 contributions (UZH)	C. Fu
0.3e	7/6/2020	Section 3.10 contributions (ILS)	I. Kureshi
0.3f	8/6/2020	Section 3.6 contributions (UPRC)	H. Georgiou
0.3g	8/6/2020	Section 3.4 contributions (UPRC)	P. Tampakis
0.3h	14/6/2020	Section 3.7 contributions (UPRC)	C. Doulkeridis, P. Nikitopoulos
0.4a	14/6/2020	Section 3.x revisions & formatting (UPRC)	H. Georgiou
0.4b	16/6/2020	Sections 1.x-2.x contributions (UPRC)	H. Georgiou
0.5	18/6/2020	Revisions in Sections 1.x-2.x-3.x (UPRC)	E. Chondrodima, H. Georgiou, P. Tampakis
0.6	19/6/2020	Completed draft for review (UPRC)	H. Georgiou
0.7	23/6/2020	Internal review completed	T. Staykova
0.8	25/6/2020	Reviewer's feedback integrated (UPRC)	H. Georgiou
0.9	26/6/2020	Section 3.9 edits (ILS)	I. Kureshi
1.0	29/6/2020	Completed final version for submission	H. Georgiou

Disclaimer

Copyright message

EXECUTIVE SUMMARY

The need for novel approaches and data analytics poses severe strain on efficient transportation and planning, as the urban landscapes are evolving very fast in the Big Data era. Analysis tools and data analytics tasks can also assist the improvement of several environmental aspects and the quality of daily life as they can indicate insights regarding safety, risk assessment. The Big Data Analytics (BDA) Toolbox of Track&Know enables us to better understand key mobility factors, which drive automotive transportation or serve as hubs for accessibility to the respective urban zones of interest. They also enable us to characterize traffic flows, regular routes, drivers' behavioural patterns in the space and time, as well as their relationship and mutual interaction with the contextual environment.

This deliverable D4.1 concludes and reports the products of Task 4.1 regarding the 'Analytics for mobility patterns detection and forecasting'. More specifically, the main challenge of BDA is to identify the real-world challenges that incorporate both the 'big data' and the 'data analytics' aspects, establish specific top-level functionalities that are currently sub-optimal or unavailable and then develop specific solutions for each one of them. At the same time, the BDA Toolbox must be as generic, autonomous and modular as possible, in order to be easily applicable to other BDA contexts and dataset modalities not included in Track&Know, e.g., from the maritime or the aviation domain, with as little changes as possible.

Based on these challenges, the BDA Toolbox consists of various task-specific components that are also generic enough to be easily applied to a much larger context than Track&Know. Each component formalizes a specific sub-task, e.g. discovering common routes or personalized trips from historic trajectory data, and provides a solid theoretical framework on how this problem can be addressed, designing beyond-state-of-the-art algorithms and methods. The developed modules are implemented inherently for the big data context and extensive experimental work is conducted to validate and optimize the models. Furthermore, the components are integrated for online or offline deployment in the context of Track&Know platform for further experiments and for supporting the project's demonstrators. More details are provided in section 1.4 with the structure of this report, as well as in section 2.2 with a brief summary of the BDA components and the functionalities they provide.

The presentation of the BDA components show that the Toolbox achieves a closely collaborating set of modules that provide top-level functionalities, ranging from customized pre-processing and data restoration & transformation per-component to trajectory analytics & clustering, discovery of mobility networks & personalized trips, hotspot analysis, driver behaviour profiling, traffic flow dynamics, etc. As this deliverable D4.1 presents in detail, the Task 4.1 requirements are met and at the same time the BDA Toolkit is versatile and modular enough that it can be employed to a much wider range of challenges than what is included in the scope of Track&Know.

TABLE OF CONTENTS

TERMS & ABBREVIATIONS

ALS	Approximation Line Segment
BDA	Big Data Analytics toolbox
BDP	Big Data Processing toolbox
BSON	Binary encoding of JSON-like documents
CER	Complex Event Recognition toolbox
CPU	Central Processing Unit
CSV	Comma-Separated Values
DP	Douglas-Pecker
GPS	Global Positioning System
HDFS	Hadoop File System
JSON	JavaScript Object Notation
MBR	Minimum Bounding Rectangle
NoSQL	Non SQL or Non Relational
OSM	Open Street Maps
PAP	Papworth
POI	Point-of-interest
RAM	Random Access Memory
RDBMS	Relational Database Management System
RDD	Resilient Distributed Dataset
SIS	Sistematica
SL	Spatial Length
SSH	Secure Shell
STR	Sort-Tile Recursive
TRL	Technology Readiness Level
VFI	Vodafone Innovus
VM	Virtual Machine
WGS	World Geodetic System
WKT	Well-known Text
XML	eXtended Markup Language

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Big Data Analytics (BDA) Toolbox harvests huge volumes of trajectories from Floating Car Data (FCD). When these data are considered in the context of urban environments, analysis methods can facilitate the extraction of useful knowledge and the provision of sophisticated services towards vehicles' drivers, citizens, stakeholders and city operators. The outcome of these methodologies can be exploited and combined with contextual information to automatically detect event occurrences within a system or a modelled domain, known as Complex Event Recognition (CER). An event recognition system accepts streams and archival data of low-level events (for example sensor data) and uses predefined patterns to recognize high-level events of interest, that is, events that satisfy/match these patterns. As an example, consider the real-time detection of vehicle/driver behavioural profile while in a specific type of urban area (e.g. city center), or events related to traffic in areas at different hours (high-level events), via combining low-level driving events or mobility patterns originating from in-vehicle or static sensors and probes.

Work package 4 introduces novel methodologies within Toolboxes which enable citizens and stakeholders to draw useful conclusions regarding the spatio-temporal distribution of traffic flows and mobility patterns using data gathered and fused from a wide variety of sources. It also addresses the issue of recognizing different classes which are formed in a data-driven way by analyzing trajectory analytics, events, accidents, etc., thus improving situational awareness and risk assessment. Using these methods and tools, forecasting models will be designed for the short- and long-term modalities that describe these mobility patterns in the future.

1.1 Purpose and scope

This task develops customized data analysis methods and tools over Big Mobility Data, including cluster analysis and motion pattern detection, by exploiting enriched and integrated data from multiple sources. Furthermore, it develops algorithms for short- and long-term forecasting of routes, flows, concentration nodes, as well as contextual characteristics, supporting outlier detection, taking advantage of previous data analytics results, as well as complex events produced within Work package 4.

As a result, this deliverable describes in detail the framework of BDA and the Toolbox that was developed. More specifically, D4.1 analyses the specification and prototype implementation of: (a) customized data analysis methods and tools over Big Mobility Data, including cluster analysis and motion pattern detection, by exploiting enriched and integrated data from multiple sources; and (b) algorithms for short- and long-term forecasting of routes, flows, concentration

nodes, as well as contextual characteristics, supporting outlier detection, taking advantage of previous data analytics results, recognizing driving behaviour and traffic flow hotspots. The core capabilities of the tools and the application of the algorithms developed, are demonstrated through a series of data-driven experimental work, closely related to the Track&Know integrated platform and its use in the project's pilots.

Deliverable D4.1 is submitted on month M30 of the project, documenting the full design and implementation details of all the BDA components, while the Toolkit is integrated in the Track&Know platform and ready to be used by the project's pilots.

1.2 Approach for the Work package and relation to other Deliverables

1.2.1 Approach & Methodology

Work package is responsible for designing, implementing and evaluating all the BDA-related components of Track&Know, including pre-processing of streaming data, exploiting enrichments (e.g. local weather), training analytical and predictive models, identifying information-rich vehicle mobility attributes and producing higher-level reasoning in relation to specific situations (e.g. risk of accident).

Several challenges are addressed by each component, depending on the type of data modalities used and the exact target of each sub-task. Trajectory analytics provide efficient segmentation and automatic extraction of locations of interest for individuals. This context is extended to analyze and investigate the optimal trip planning for electric vehicles. Clustering is also employed in trajectory analytics, in order to identify common routes and sub-segments. Predictive analytics is investigated via two different and supplementary approaches: (a) a deep learning method for robust 'blind' detection of mobility patterns in the short-term; and (b) a clustering-based method that predicts the evolution of vehicle trajectories in the long-term based on common routes. For driver behaviour characterization, low-resolution GPS data and context-aware enrichments (local speed limits) are pre-processed, augmented via robust feature encoders and categorized via unsupervised models in a purely data-driven way. Additionally, the detection of hotspots, i.e., nodes of high spatio-temporal density of traffic, are detected dynamically via mobility flow analysis, while other methods are also applied for activity-based pattern detection, semantic enrichment analysis and genetic p-median solver for location-allocation problems.

The BDA Toolbox was designed from the beginning as a collection of autonomous yet easily interconnected components that address various tasks in the general context of data analytics for the project's big data challenges. Although each component has its own internal structure and processing pipeline, they all exploit parts of the processing and output that is produced by the BDP Toolkit, which is the main producer of the input data feeds here. As such, each of the BDA components was designed, developed and tested upon different Track&Know sub-tasks

and datasets, which are presented separately in detail in the corresponding sub-sections. This modular approach enables the application of these components selectively, combined for specific purposes and problems at hand, not only in the context of Track&Know but in any similar BDA challenges.

1.2.2 Relation to other deliverables

The work described in this deliverable relates to previous deliverables of the project, as follows:

- D1.2 *'Corporate Big Data Requirements'*: The specification of requirements regarding storage and querying of big data for supporting the needs of advanced data analysis operations has guided some design decisions made in this deliverable.
- D2.1 *'Architectures for the management of structured & unstructured data streams'*: The solutions described in this deliverable are compliant with the overall architecture of Track&Know.
- D2.2 *'Architectures for the Management of Batch and Interactive Data Sources'*: Includes the comparative overview of different NoSQL solutions and the selection of a document-oriented store (MongoDB) to be the basis of the batch storage solution in Track&Know.
- D3.1 *"Data Acquisition and Integration Report"*: This report essentially describes the process that enriches mobility data by means of cleansing, map-matching, and associating GPS traces with weather and points-of-interest (POIs). The resulting enriched mobility data comprises the input data set for persistent and scalable storage that is considered in this deliverable.

Additionally, this deliverable D4.1 is combined with deliverables D4.2 *'Analytics for individuals' mobility networks'* that describes another set of methods and techniques for creating aggregated mobility models, D4.3 *'Transfer learning for mobility data models'* and D4.4 *'Analytics for complex event recognition'* that conclude the rest of the Work package 4 work, as well as deliverables D5.1 *'Visual Analytics for Big Mobility Data'* and D5.2 *'Visual Analytics for complex events'* that explore the presentation and visual analytics aspects of the Work package 4 outputs. Finally, the BDA Toolbox components and their integration in the Track&Know platform are associated with the three Demonstrator deliverables, i.e., D6.2 *'Demonstrator 1 - Auto-Insurance & Innovative Mobility Services'*, D6.3 *'Demonstrator 2 - Emergency Healthcare'* and D6.4 *'Demonstrator 3 - Fleet Management'*, as described appropriately for individual components.

1.3 Mapping Track and Know outputs

The following list describes how the Grant Agreement commitments are addressed in this report, within both the formal Deliverable and Task description, regarding the work performed and the project's expected outputs.

- (T4.1/D4.1) *'...develops customized data analysis methods and tools over Big Mobility Data...'*: Section 3, with each sub-section describing an individual sub-task and the component developed to address it.
- (T4.1/D4.1) *'...cluster analysis and motion pattern detection...'*: Section 3.3 for sub-trajectory clustering, as well as sections 3.1 and 3.2 on how these can be applied for personalized trips characterization and energy use optimization.
- (T4.1/D4.1) *'...exploiting enriched and integrated data from multiple sources...'*: Section 3, with each component describing the exploitation of various data enrichments provided by the BDP pipeline, e.g., as in section 3.8.
- (T4.1/D4.1) *'...algorithms for short- and long-term forecasting of routes...'*: Section 3.4, with part-I describing the short-term approach (NN-based) and part-II describing the long-term approach (routes-based).
- (T4.1/D4.1) *'...flows, concentration nodes, as well as contextual characteristics, supporting outlier detection...'*: Section 3.6 for hot spot analysis, section 3.7 for activity-travel patterns.
- (T4.1/D4.1) *'...previous data analytics results, as well as complex events...'*: Section 3.5 for driver behaviour profiling.
- (D4.1) *'...core capabilities of the tools and the application of the algorithms developed, will be demonstrated through an open research data pilot...'*: Section 3, all component descriptions include experimental protocols and results that are used as guidelines for setting up the corresponding demostrators for D6.2, D6.3 and D6.4.
- (D4.1) *'...report will include several use cases to allow the use and extension of the tools/algorithms...'*: Section 3, all component descriptions include experimental protocols and results that are used as guidelines for setting up the corresponding demonstrators for D6.2, D6.3 and D6.4.

1.4 Structure of the deliverable

The structure of this deliverable is built around the core material of the BDA Toolbox, i.e., the description of each individual component and how it relates to specific tasks in Work package 4 of Track&Know.

Section 2 provides a few comments about how the BDA Toolbox and individual components relate to the overall Track&Know platform. Section 3 includes the core material for the BDA Toolbox. Specifically, section 3.1 is for the extraction of individual locations for interest; section 3.2 is for the analysis of electrificability of trips; section 3.3 is for distributed sub-trajectory clustering and its applications; section 3.4 is for Future Location Prediction (FLP) consisting of two parts, part-I for the short-term context (NN-based) and part-II for the long-term context (routes-based); section 3.5 is for driver behaviour profiling based on trajectory analytics; section 3.6 is for hotspot analysis and its applications; section 3.7 is for the identification of business activity-travel patterns; section 3.8 is for the exploitation of semantic enrichments of trajectories for cross-scale analytics; and section 3.9 presents a p-Median solver based on genetic algorithm for location-allocation tasks. Section 4 concludes the report with some additional comments and insights.

2 Relevance to the Track and Know platform

2.1 Track and Know platform at a glance

As presented in detail in deliverable D2.1, the Track&Know Big Data Platform consists of:

- Data sources, which represent the structured and unstructured data streams to be made available and be connected to the platform;
- Data store, which represents the batch and interactive data sources that will be made available and will be connected to the platform;
- Connectors, together with the Communication platform, that connect external Data sources and the Data store and make them available to the platform, Toolboxes and Pilots;
- Underlying Infrastructure providing all the necessary Big data tools.

The Track&Know high-level architecture is illustrated in Figure 1 (source: D2.1).

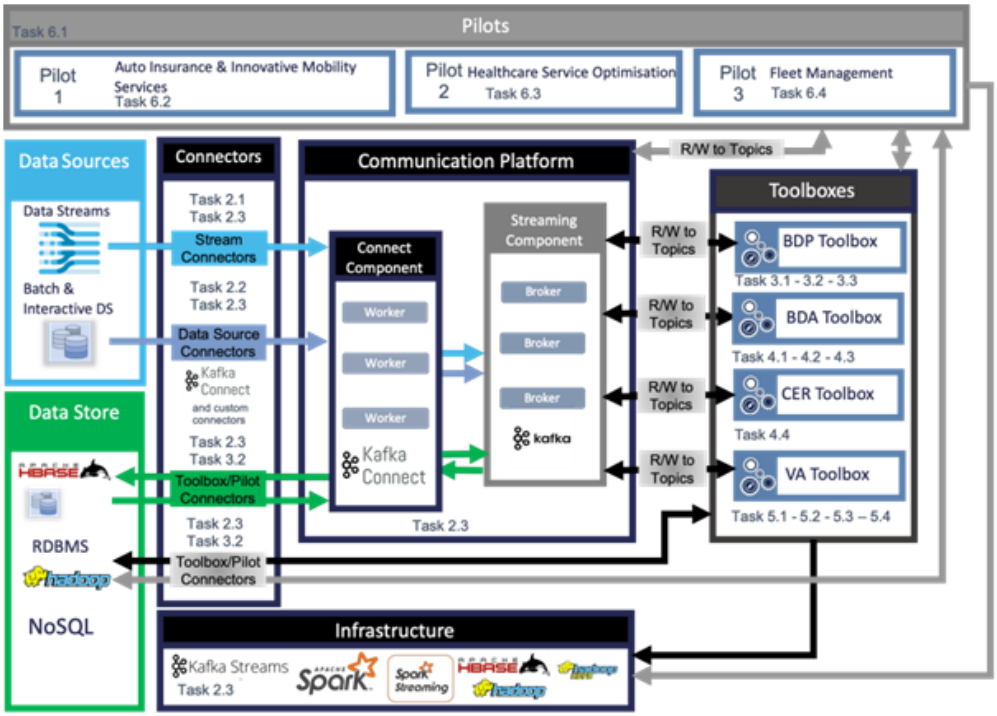

Figure 1: Overview of the Track&Know Big Data Platform.

2.2 The Big Data Analytics (BDA) Toolbox

As described earlier, the BDA Toolbox is a collection of autonomous and closely collaborating components that address specific sub-tasks of Work package 4, more specifically the BDA-related functionalities of the Track&Know platform, summarized below - specific sections in 3.x provide the details:

Extraction of Individual Locations of Interest: Given the GPS traces of a single mobility user, this component identifies a trajectory segmentation based on individual stop distributions, then performs a parameter-free spatial clustering of the start- and end-points, which represent the locations that the user visited. Each output location is represented by a set or points and its representative centroid. The component is realized in Python as a hybrid of X-means (a centroid-based clustering method) and single linkage agglomerative clustering with a statistics-based halt condition. The efficiency and the lack of parameters make this method completely automatizes and suitable for working (sequentially or in full parallelism) over large sets of users' data. Each user is processed independently. The expected input is a set of GPS traces of an individual.

Construction of Individual Mobility Networks (IMNs): This component builds a network representation of the mobility history of a user. Nodes represent user's Individual Locations of Interest and edges represent the trips between two locations. Nodes and edges are enriched with several statistics of the associated trips, such as temporal distributions and distances. The component is realized in Python and completely automatized. Each user is processed independently, based on a set of reconstructed trips as input.

Semantic Annotation of IMNs: This component analyzes the mobility history and the IMNs of several users in order to characterize locations through mobility and contextual features, then classifying locations into categories that are used to group individual locations of a user into higher-level locations. The component is realized in Python and completely automatized. Each user is processed independently. The primary objective of this component is to make IMNs applicable to trip data different from private cars.

Extraction of mobility-based city indicators for transfer learning: Given a specific geographical area, this component analyzes the corresponding road network structure and actual mobility data to extract a list of several indicators that characterize the area. The indicators currently include measures describing five types of phenomena: population concentration; traffic flows; Individual Mobility Networks (see component above); road network structure; traffic concentration on the road network. The component is realized in Python and completely automatized.

Analysis of electrificability of trips: Given the mobility data of a trip, the component estimates the point-to-point charge consumption that an electric vehicle would require to perform it. Given the trips of a user for a whole day and the initial charge level, this can be used also

to check if her daily mobility might fit an electric vehicle without the need of recharging. The component is implemented in Java and completely automatized, and takes into consideration various factors: speed, accelerations (currently inferred from GPS speed) and steepness (currently inferred from public ground elevation models).

Trip planning and simulation for electric vehicles: This components finds the fastest path between two points of a road network taking into consideration the limitations of an electric vehicles, in particular: computes an estimate of battery consumption during the trip; where needed, the trip will include a stop at a recharge station before battery depletion; the recharge time is simulated and added to the overall travel time. The component is implemented in Python and includes a (partially automatized) phase for the reconstruction of a road network enriched with information about speeds, elevations and (derived from them) battery consumption; and a (fully automatized) phase that computes the optimal path. The input consists of starting and ending coordinates of the trip, and the starting battery level. The output includes the route, the recharge points, the ending battery level and the travel time.

Individual long-term event risk prediction: The problem addressed consists in building a predictive model able to assign a probability of some long-term event happens to an individual, based on the computation of mobility features describing the user. The component combines other ones listed above, in particular individual locations and IMNs, and on top of them builds features of various types based on the mobility history of the user: mobility demand, driving style, changes in mobility, mobility context. The features are correlated to events that happen in the mid/long-term future, such as crashed in the next month, through a machine learning predictive model. The component is implemented in Python, is completely automatized and takes as input the mobility data of the user in the monitoring period, as well as (only for the construction of the model, and not for its usage in prediction) a label specifying whether the event is going to happen.

Traffic prediction in new urban areas: This method implements a network-based ML approach to predict (or simply estimate) the traffic flow in urban geographical areas, exploiting the road network structure, traffic flows in the neighborhood, and various open datasets about the same area. The component is realized in Python and is partially automatized, as it requires some data collection and preprocessing. The required input is the OD matrix between known areas (obtained from trajectory data) and the list of POIs and Airbnb items with their features. The output are the expected in- and out-flows between a new area (whose connection with the existing ones is known) and all the others.

Distributed Sub-trajectory Clustering: The problem of Distributed Sub-trajectory Clustering (DSC) lies in identifying groups of moving objects that moved 'close' enough in space and time for at least some time duration, and whatever is the method it should be performed in a distributed way (for scalability purposes). The proposed approach consists of three steps. The first step is to retrieve for each trajectory, all the moving objects, with their respective portion of movement, that moved 'close' enough in space and time for at least some time duration. The

second step takes as input the result of the first step and aims at segmenting each trajectory into a set of sub-trajectories. The way that a trajectory is segmented into sub-trajectories is neighborhood-aware, meaning that a trajectory will be segmented every time its neighborhood changes 'significantly' so as to result to homogeneous sub-trajectories (w.r.t. their surrounding moving objects). The above procedure will result to a set of sub-trajectories. In the third step the goal is to create groups (whose cardinality is unknown) of similar sub-trajectories and at the same time identify sub-trajectories that are significantly dissimilar from the others (called outliers).

Future Location Prediction (FLP) - Trajectory Prediction (TP): The task of Trajectory Prediction (TP) can be addressed via two distinct approaches. The first, 'network-agnostic' method assumes nothing about the underlying road network and exploits only a) patterns made available by previous locations of various vehicles, and b) the very recent history of previous locations, in order to predict the evolution of the vehicle's future positions via Long Short-Term Memory (LSTM) models for the purposes of Future Location Prediction (FLP). Note that this method is able to predict future positions of vehicles with unknown historical trajectories in the LSTM training phase, i.e., not included in the historical data used in training, as long as they manifest more or less similar movement patterns. The input can be either the raw GPS (noise-filtered) or their map matched counterparts (see BDP data pipeline module above). Also, the map matching component can be employed to align the LSTM predicted locations as a post-processing step. The second, 'network-aware' method exploits an underlying road network of 'discovered' as medoids via clustering and the evolution of the vehicle's trajectory is based on the map matching in a probabilistic sense, i.e., select the most probable route based on previous history (training data). Both approaches can be enabled for batch or online modes.

Driver Behavior Profiling (DBP): The Driver Behavior Profiling (DBP) component implements a data-driven approach to the challenge of analyzing, encoding and classifying driver behavioral patterns in the short- or the long-term, which in principle are of unknown categories. Multiple time series of moving vehicles, i.e, speed & acceleration in addition to location, are tracked and analyzed, statistical features are calculated and then used as input for training classification models. In the case when specific driver categories are not provided as ground truth, unsupervised learning (e.g. clustering) can be used instead. The trained models can then be used in batch or online modes, in order to characterize the general driver behavior (long-term) or tracking it as the vehicle moves (short-term). Previous work was on acquiring the state-of-the-art in literature, which seems to be focused mostly around high-quality, high-rate data of location, speed and acceleration. Other approaches require modalities outside the scope of Track&Know, e.g. visual tracking of the driver's face, proximity sensors installed on the vehicle, etc. Regarding trajectory data, there are approaches using statistical or spectral features for analyzing the data series and detecting abnormalities. Additionally, the proposed approach is designed for online/streaming mode and lightweight yet powerful analytics, in order to be applicable to on-the-fly driver behavior profiling. A dynamic temporal resampling algorithm is employed for

transforming the sparse, variable-rate, GPS-only trajectory data into three distinct location-invariant time series, namely speed, acceleration and turn rate, after the raw GPS trajectories are map-matched to the underlying road network and noise-filtered for removal of artifacts. A wide range of statistical, time series and spectral methods are implemented as feature functions or 'encoders' of various aspects of short-term mobility tracking. The results show that such an approach is feasible, despite this challenging context of constraints, providing a data-driven adaptive way to recognizing 'normal' and 'abnormal' driving patterns on-the-fly.

Hot spot Analysis: Hot spot analysis is the problem of identifying statistically significant spatial clusters from an underlying data set. We have developed a parallel and scalable algorithm (THS) for trajectory hot spot analysis, using an adapted version of the Getis-Ord statistic tailored for trajectory data rather than point data. In brief, we split the 3D spatio-temporal space in cells of user-defined granularity and we map the positions of moving objects to cells. Then, we perform parallel processing of these cells to compute the Getis-Ord statistic for each cell, and we are able to output the top-k cells according to their Getis-Ord value. Our algorithm is developed in Apache Spark.

Identifying business activity-travel patterns based on GPS data: As employers, suppliers and transport providers, organizations are responsible for the generation of a large part of traffic flows on the transport network. However, despite the significance of business travel to overall mobility, the underlying activity compositions of the movement and the decision making process within organizations have not been well understood. To address this challenge, a new method has been developed in this toolbox, aimed to identify typical activity-travel patterns from business trips and characterize travel behavior of specific companies or vehicles (and corresponding drivers) based on the obtained patterns. The method and derived results will help uncover business activities and travel features, providing an improved behavioral mobility understanding, and exploring factors that would lead to addressing the increasing challenges related to business travel (e.g. environmental issues, driving safety, and travel demand management). .

Semantic Enrichment of Trajectory for Cross-Scale Analysis: In this component, we will investigate how trajectories can be semantically enriched with geospatial context (e.g., POIs, road network, geographical events, land use, weather) to allow cross-scale analysis in the spatial, temporal and thematic dimensions. This is based on the observation that mobility behaviors may manifest different movement patterns at different scales in the spatial, temporal, or thematic domain. The main focus here is to make the component scalable to different amounts of data (i.e., volume of big data). Spark and Python are adopted in the implementation.

Genetic p-Median solver: This component uses mobility information to infer demand points and make location-allocation decisions for facilities. The BDA Toolbox implementation uses a Spark and node-level parallelised Genetic Algorithm approach to solve the p-Median problem. The tool is able to solve this NP-hard problem in polynomial time.

3 Big Data Analytics (BDA) components

A brief description of the BDA Toolbox was summarized in section 2.2, providing insights on what is included and what top-level functionalities are available ad-hoc for addressing specific sub-tasks. In this section, every individual BDA components is described in detail, including the specific problem formulation, the methodology employed, the experimental protocol and the datasets used, as well as the results and the outcomes from the approach.

3.1 Adaptive Extraction of Individual Locations of Interest

In summary:

- *Generic question addressed*: Identify sub-groups of stop points.
- *Track&Know specific question*: Identify the relevant locations of an individual user.
- *Novelty / Advantage over existing methods*: Does not require stop duration thresholds, and provides more stable locations.
- *Experiments conducted*: Testing on a Track&Know Pilot dataset with quantitative and qualitative evaluations.
- *Type of analytics*: Descriptive analytics.
- *Automation / TRL*: TRL 4 (automated tool, tested on real data for simulation-based applications)
- *Extension to other domains*: None; the tool is specific for vehicle mobility data.

Users' locations detection is a basic task for many applications and analyses related to the GPS mobility data of individuals. Its objective is to identify the areas where each user performs her activities, based on the analysis of the places (essentially, GPS points) where she stopped. Examples of locations are home, the work place, a supermarket, a gym, a fuel-station, etc. In literature this problem is typically addressed by first identifying the user's stops with some simple heuristics, and then applying generic clustering algorithms which are able to group the user's stop observations by means of some distance function. Thus, the resulting clusters will be interpreted as user's locations. Figure 2 shows a fictitious example of all the process.

However, existing solutions suffer from various drawbacks. First, stops are typically defined based on fixed thresholds that describe when the user performs a significant stop, under the assumption that all users can fit the same threshold values. As we will discuss in this section,

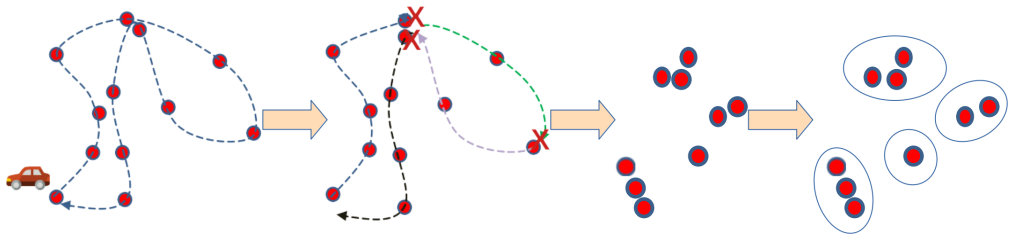

Figure 2: Example of location extraction process: the initial sequence of points is segmented into trips and stops, which are then clustered to identify locations

such assumption can be handy for first-cut solutions, yet they might miss stops that are significant for the single individual yet not fitting the global thresholds. The solution we provide on this line consists in an automatic procedure to find custom thresholds for each user, thus adapting to the individual behaviour. Second, the stop clustering phase is sometimes focused on specific optimization criteria, such as compactness maximization or density connectivity, that not always correspond perfectly to the notion of locations, and therefore the results (though optimal w.r.t. its own criteria) are not good locations for mobility applications. To this purpose, we adopt an ad hoc approach called TOSCA [66], adapted to integrate the stop detection method mentioned above. TOSCA is a parameter-free solution that, again, automatically adapts to the individual data, and whose good scalability performances make it a perfect fit for the big data applications tackled within the Track&Know project.

In this section we will introduce our new adaptive stop detection schema, and discuss the stop clustering solution adopted. Finally, evaluations will be provided both in terms of stop detection (compared to standard approaches) and in terms of impact on the final location detection phase. Indeed, the improved procedure introduced here, typically allows to identify a larger set of locations for the user, since for some of them it is possible to lower the thresholds and discover individually-significant stops.

3.1.1 Trajectory segmentation

In mobility analytics one of the fundamental concepts is *movement*, meaning with that the part of mobility data that describes a transfer from one place where the individual (or the object) was staying, to another one were the user will stop. Identifying movements in the raw stream of positions, for instance the continuous flow of GPS traces of a vehicle, is essential yet non-trivial. While it is simple to define a *stop* in geometrical terms, it is much less clear how to define *significant stops*, i.e. stops that might have some meaning for the user (for instance, stopping to do some activity before leaving), as opposed to stops that are simply incidental (for instance, due to a small traffic jam).

Practitioners in the mobility analytics domain defined several simple strategies to select stops in the mobility data stream (a brief account of literature on this topic is provided in the next section), each of them apparently capturing well some specific concept or some application-specific

idea of stop. For instance, some solutions simply identify the moments where the object did not move, based on some thresholds, while others select the stops that have a duration compatible with some specific task, for instance discarding stops at a supermarket if their duration is physically too short to be able to enter, buy and exit. However, most existing solutions suffer from two important limitations: (i) they are based on critical thresholds that the user needs to choose accurately, and in most cases it is difficult to understand what value is the best; (ii) such thresholds are global, i.e. the same threshold value applies to all the moving individuals, irrespective of any distinctive characteristics they might have. The reason of the latter is that, while an overall evaluation might be performed to guide the choice of a global threshold, doing that separately for each individual might be impossible if their number is huge.

In our work we try to overcome the limitations highlighted above, providing a general methodology that inspects the mobility of the individual, and identifies segmentation thresholds that apparently match her mobility features. The process allows to get rid of any input parameter, adapts thresholds to each single individual and, most importantly, is completely automatic, thus applicable to large pools of users.

3.1.2 Related Work

Segmentation is a technique for decomposing a given sequence into homogeneous and possibly meaningful pieces, or *segments* such that the data in each segment describe a simple event or structure. Segmentation methods are widely used for extracting structures from sequences, and are applied in a large variety of contexts [177]: time series [73, 24], genomic sequences [102, 139, 153], and text [96], to cite a few.

The segmentation of human trajectories is a very valuable task as it enables the development of mobility data models [155, 64] and applications like carpooling [63], or trajectory prediction [180]. Various simple approaches are currently adopted in practice. In [179] human trajectories are extracted adopting a predefined rule based on a pair of spatio-temporal parameters regulating the end of a trajectory and the start of the subsequent one. Similarly, in [68] the trajectory is divided into subsequent trips if the time interval of "nonmovement" exceeds a certain threshold. In [203] it is described a change-point-based segmentation approach for GPS trajectories according to the transportation means adopting a universal threshold for determining whether a segment is "walk" or "nonwalk". The work in [23] presents a theoretical framework that computes an optimal segmentation by using several criteria (e.g., speed, direction, location disk) that are satisfied in each partition, thus making the approach local, and applied computational geometry methods. However, their methods are general and not clearly applicable to the human trajectory context, where a trip can be complex and not show the geometrical/movement uniformity the methods look for. Finally, each criterion corresponds to thresholds that the user must set, without clear guidelines on how to choose them.

The authors of [196] segment the trajectories in two steps. The first segmentation is performed by means of simple policies with respect to temporal and/or spatial predefined constraints. Then,

the trajectories are divided into *stops* and *moves* observing variations of the speed of the object. If the variations of the speed is below a speed threshold and there is a sufficient number of observations, then the portion of trajectory is annotated as a stop. The speed threshold is not general but changes according to the user behavior and also to the surrounding of the stop. In [164] is defined a measure of the density of the points in the neighbourhood of each trajectory point, the Spatio-Temporal Kernel Window (STKW) statistics. To determine the start and end points of segments, the algorithm looks for maximal changes in STKW values. The focus of the approach is on capturing changes of transportation mode, including stops, which are simply points with low speed.

In addition to those mentioned above, several other solutions to the trajectory segmentation problem have been proposed in literature, yet with objectives different from ours. For example, cost-function based strategies were presented in [87][86], while clustering-based ones are introduced in [97] [101]. All these approaches are focused on splitting a movement into homogeneous parts, rather than discovering significant stops, which is the purpose of this paper.

In our solution we provide a segmentation method that, opposed to most of the approaches mentioned above, is not based on fixed space and/or time thresholds to be fixed by the user – this is the case, for instance, of [179, 196, 68, 203]. Instead, we aim to make the segmentation parameter-free and also adaptive to the single user's data, giving the opportunity to have different kinds of segmentation over different users. Also, our approach is complementary to the STKW-based one [164], as the latter aims to differentiate movements with different speed profiles, including stops as a particular example, while we focus on stop timing and try to understand which stops are actually significant (e.g. not too short) for the user. A similar work was proposed in [41]. Here the authors proposed a new approach called Octal Window Segmentation (OWS) for unsupervised trajectory segmentation. The intuition behind their approach is that when a moving object changes behavior, this shift may be detected using only its geolocation over time. So the work focuses on finding these changes only from the object's coordinates using interpolation methods to generate an error signal. This error signal is then used as a criterion to split the trajectories into sub-trajectories.

3.1.3 Problem definition

We start by defining trajectory segmentation based on a spatial and a temporal threshold, in a way similar to standard approaches in literature.

Definition 1 (Individual trajectory) *Given a user u, her Individual Trajectory T_u is the sequence of n points $T_u = \langle p_1, \ldots, p_n \rangle$ that describes her position in time, where each point $p \in T_u$ is defined as a triple $p = (p.x, p.y, p.t)$, representing its spatial coordinates x and y and the corresponding timestamp t. Moreover, points are in chronological order, i.e. $\forall 1 < i \leq n . p_{i-1}.t < p_i.t$.*

Definition 2 (Pseudo-stop duration) *Given an individual trajectory $T = \langle p_1, \ldots, p_n \rangle$ and*

a spatial threshold σ, the Pseudo-stop duration associated to point p_i is defined as $SD(T,i) = \min\{p_j.t - p_i.t | i < j \leq n \wedge d(p_i,p_j) > \sigma\}$, where d represents the geometrical Euclidean or geographical distance.

Notice that the last point p_n will have $SD(T,n) = \min \emptyset = \infty$.

Definition 3 (Segmented trajectory) *Given a trajectory $T = \langle p_1, \ldots, p_n \rangle$, a spatial threshold σ and a temporal threshold τ, we define the (σ,τ)-segmentation of T as $T^{\sigma,\tau} = \langle S_1, \ldots, S_m \rangle$, such that:*

$$(i) \quad \forall 1 \leq i \leq m. \exists 1 \leq s < e \leq n : S_i = \langle p_s, p_{s+1}, \ldots, p_e \rangle$$

$$(ii) \quad \bigcup_{i=1}^{m} set(S_i) = set(T)$$

$$(iii) \quad \forall 1 \leq i \leq m. \forall 1 \leq j \leq |S_i| : SD(S_i, j) > \tau \Leftrightarrow j = |S_i|$$

$$(iv) \quad \forall 1 \leq i \leq m. S_i \text{ is maximal}$$

where $set(I) = \{p \in I\}$.

Conditions (i) and (ii) imply that the segments of the segmented trajectory of T form a partitioning of the elements of T in the strictly mathematical sense. Moreover, conditions (iii) and (iv) state that all the points in a segment are movement points, i.e. their pseudo-stop duration is smaller than the given threshold, excepted the last point. Therefore, each point in T that has a high pseudo-stop duration will act as a split point, and corresponds to a distinct partition in $T^{\sigma,\tau}$.

Existing trajectory segmentation methods assume that the same rules and the same parameters should apply to all moving objects. Since different objects can show very different movement characteristics, the above assumption leads to make choices that on average fit best the dataset, yet potentially making sub-optimal choices on single individuals.

Our objective is to overcome this limitation, making the segmentation process adaptive to the individual and taking into consideration her overall mobility. Our problem statement extends the traditional formulation of segmentation as a threshold-based operation, thus the core issue is to find good parameter values for each user.

Definition 4 (Individual cut threshold problem) *Given an Individual Trajectory T_u, and a global spatial threshold σ, the problem is to identify the temporal threshold τ that yields the optimal segmentation $T^{\sigma,\tau}$.*

Since the number of moving objects can be very large, the process must be completely automatized and require no human intervention. In Section 3.1.4 we will introduce a simple and effective approach to solve the problem and thus find a suitable value of τ for each user. In addition, some basic guidelines to choose a value for the global spatial parameter will be provided.

3.1.4 Self-Adaptive Trajectory Segmentation

The proposed solution to the individual cut threshold problem consists in fixing the spatial threshold to a global value (i.e. to be used for all users) and then in studying the segmentations that we would obtain by applying different temporal thresholds. We will start describing the process for choosing the temporal threshold, which is the central part of the solution, and later discuss how the spatial one can be chosen.

When very small values of τ are used, the segmentation obtained will contain a huge number of very short segments, till the extreme case where each point forms its own segment. As the threshold is increased, more and more segments will merge together, since some of the former splitting points will fall below τ. The process is expected to gradually enlarge the trajectory segments by first including simple slowdowns (i.e. not really stop points), then temporary stops (e.g. at traffic lights), and so on.

Our approach consists in (virtually) monitoring such progression, and detect the moment where an anomalous increase in the number of segments is observed, which represents a sort of *change of state*. This follows the same kind of idea adopted in various unsupervised classification contexts, such as the *knee method* for deciding the number k of clusters for the k-means algorithm, or analogous solutions to choose the radius for density-based clustering (e.g. DBScan).

In our solution, rather than relying on visual or similar heuristic criteria, we will base the threshold selection on a statistical test. In particular, we will adopt the Modified Thompson Tau Test [22] which, basically, checks whether a given value fits the distribution of the rest of the data or not. Since we look for anomalous values in a sequence, we apply the test iteratively, comparing each value $n(t)$ (the number of segments obtained with $\tau = t$) against the values $n(t')$ obtained for larger thresholds t'.

This process yields a set of thresholds that have an anomalous number of partitions as compared to the successive thresholds. Among them, we simply choose the smallest one, thus deciding to select the segments that emerge at the first *change of state*, also representing shorter and finer granularity movements.

The procedure, named ATS (self-Adaptive Trajectory Segmentation) is summarized in Algorithm 1. Step 3 collects the pseudo-stop durations SD of all the points i that make up the segment, and step 4 computes the frequency F of each value, basically representing the number of new segments obtained using that value as τ w.r.t. the previous smaller thresholds. In our implementation such frequency distribution is computed through smoothed histograms, grouping values into bins of 1-minute width. Figure 3(left) shows the frequency distribution of a sample trajectory, the vertical line corresponding to a possible cut point. The resulting set of segments obtained is described in Figure 3(right) in terms of segments duration. Finally, step 5 selects the frequency values that appear to be anomalous (based on the Modified Thompson Tau Test) w.r.t. the frequency of larger thresholds, and step 6 returns the earliest time threshold that has an anomalous frequency.

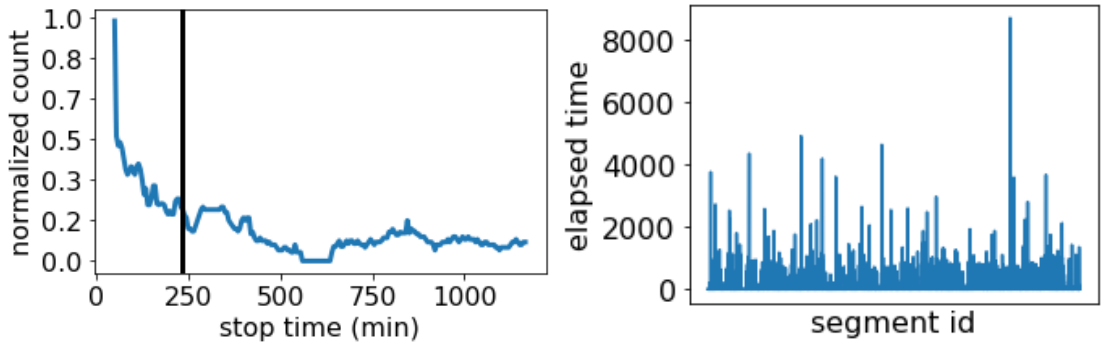

Figure 3: Frequency distribution of pseudo-stop durations for a user trajetory (left), and the durations of the segments obtained using a specific threshold to cut the trajectory (right). The threshold used corresponds to the vertical line on the left image.

Computational complexity

The cost of Algorithm 1 is dominated by step 3, since the computation of each pseudo-stop duration (SD) could in principle require to scan all the remaining points of the individual trajectory, thus yielding a $O(n^2)$ cost, where n is the size of the individual trajectory. However, in practical applications the trajectory portion needed for each SD is relatively small, leading to a quasi-linear cost. The remaining parts of the algorithm can be realized a linear time, including the Modified Thompson Tau Test which can be computed for each points through incremental updates.

Algorithm 1: ATS(T, σ)

1 **Input:** Individual trajectory T, spatial threshold σ
2 **Output:** Cut threshold τ
3 $S = \langle\ SD(T,i)\ |\ 1 \leq i \leq |T|\ \rangle$;
4 F = frequency distribution of S values ($F(a) = |\{a \in S\}|$);
5 $C = \{t | t \in range(F) \wedge TT(F(t), \langle F(t') | t' > t \rangle) = true\}$; // $TT(a, B)$ = *Modified Thompson Tau Test of a vs. set B*
6 **return** $\min C$

Fixing the spatial threshold

In our approach, the threshold σ represents the minimum distance between two (consecutive) points that can be considered as a movement, and the temporal parameter is indeed measured as the time needed to make a movement. A simple way to fix its value is to adopt the minimum value that, according to the accuracy of our dataset, cannot be mistaken for a positioning error, for instance due to GPS uncertainty. In our experiments we adopt road vehicle GPS traces that are expected to have errors not larger than 10 meters, therefore we could fix $\sigma = 20$ (the

worst case distance between two points that have the maximal error in opposite directions). We decided to slightly increase it to 50 in order to stay on the safe side, also to take into account that errors are slightly higher than average in urban centers, which is the application context where our experiments are performed. Since we do not have data source from other kind of transport (ships, planes, etc.) the selected threshold seems to meet our purposes. However, empirical results confirm that the value of the global parameter σ is not critical, as our approach shows a low sensitivity to it. For this reason, the value we chose in our experiments (50 meters) can be considered a good guess for generic vehicle GPS data. Other data sources with a higher spatial uncertainty might require larger values.

3.1.5 Evaluation Measures for segmentation

The reconstruction error generally used for evaluating segmentation problems [18] just measures how well each segment can be approximated with one value, and thus seems not to fit with trajectory segmentation. Therefore, similarly to clustering evaluation, we propose three internal evaluation measures [172]. Let T be the sequence of n points and $T_S = \langle S_1, \ldots, S_m \rangle$ its segmentation. We denote with $A_t = duration(T) = p_n.t - p_1.t$ the total elapsed time from the first point of $p_1 \in T$ to the last point $p_n \in T$, and $A_d = length(T) = \sum_{i=1}^{n-1} d(p_i, p_{i+1})$ the total distance covered by the trajectory, computed by considering every couple of subsequent points in T. Let $M_t = \sum_{S_i \in T_S} duration(S_i)$ be the sum of the segments' duration, i.e., the time spent driving, and $M_d = \sum_{S_i \in T_S} length(S_i)$ be the sum of the segments' length, i.e., the distance traveled. Then, we define the following measures:

- *time precision*: $TP = 1 - M_t/A_t$
- *distance coverage*: $DC = M_d/A_d$
- *mobility f-measure*: $MF_\beta = (1 + \beta^2) \cdot TP \cdot DC / ((\beta^2 \cdot TP) + DC)$

All measures range from zero to one. The higher the value the better the result. The objective of these measures is to promote segmentations capturing long stops (*time precision*) yet also covering most of the overall distance (*distance coverage*). These two objectives are conflictual, since making stops longer reduces the number of points that contribute to the distance covered. The *mobility f-measure* accounts for both aspects simultaneously. In the experiments we adopt $\beta = 0.25$, which weighs *time precision* much higher than *distance coverage* by augmenting the relevance of missing precision in stop detection. The reason is that *i)* it is relatively easy to guarantee an high distance coverage, and *ii)* the main focus of the paper is on the temporal aspects of trajectory partitioning.

3.1.6 Experiments

We experimented the proposed self-adaptive trajectory segmentation approach (ATS) described above over a real dataset of GPS vehicle traces. The results commented in the following refer to

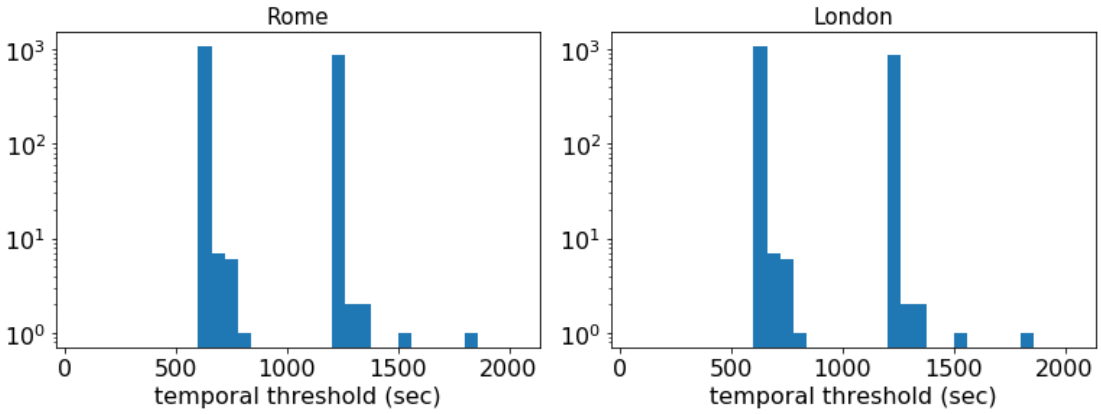

Figure 4: Time threshold distributions for trajectories in Rome and London. The peaks show the ideal thresholds to be set to build the trajectories.

2000 users of the area of Rome (Italy), and London (UK). The means and standard deviations of the sampling rate for the users analyzed are 12194.67 ± 22575.66 and 4385.76 ± 9359.14, for Rome and London respectively. The high values and their high variability is due to the presence of several long time gaps, typically due to parking periods.

In the following we first analyze the personal temporal thresholds returned by ATS, then we propose a quantitative and qualitative evaluation of the results for understanding the benefits of the novel method with respect to existing ones. We compare ATS against the trajectory segmentation method with fixed parameters proposed in [179] ($\text{FTS}_{\text{temp-thr}}$). Moreover, we adopt as baseline a random trajectory segmentation method that segments the sequence of points $T = \langle p_1, \ldots, p_n \rangle$ into m equal-length segments *(i)* with m randomly extracted between 2 and $n/2$ (RTS_1), or *(ii)* with m set to the number of segments returned by the proposed ATS method (RTS_2).

Self-Adaptive Temporal Threshold

We observe in Figure 4 the distribution of the time thresholds selected by ATS for each user (vertical axis represents value frequencies in log-scale).

Although every user has her own mobility behavior with its own mix of regular and more erratic behaviours [136], we observe two clear peaks in the distributions for both Rome and London. This means that with respect to ATS we mainly recognize two different types of users regarding to the minimum duration of the stops. This supports the intuition behind our approach, namely to have a self-adaptive procedure selecting a personalized best temporal threshold for each user. Selecting one single threshold value for all the data might negatively affect the segmentation of some users, partitioning their trajectories either too much or too little. The first peak is at about 600 seconds ($\sim$ 10 minutes), while the second peak at 1200 seconds ($\sim$ 20 minutes). These values correspond to the temporal thresholds that the ATS procedure uses to cut each trajectory.

method	$MF_{.25}$	TP	DC	$ratio_{\text{sr}}$	$\#segms\ (avg \pm std)$
ATS	**.951**	.951	.981	0.049	837.34±854.52
FTS_{120}	.925	**.996**	.456	**0.015**	592.26 ±652.78
FTS_{1200}	.948	.947	**.997**	0.053	746.28 ± 733.96
RTS_1	.279	.268	.722	0.700	2094.85± 2472.36
RTS_2	.124	.118	.877	0.883	899.59 ± 926.03

Table 1: Evaluation on Rome data. The first three columns show the measures adopted to test our new approach. The fourth one reports the ratio between the average sampling period of non-stop points over that of all points, and the last column is the number of segments.

method	MF_{25}	TP	DC	$ratio_{\text{sr}}$	$\#segms\ (avg \pm std)$
ATS	.955	.953	**.999**	0.047	433.915±513.715
FTS_{120}	**.958**	**.961**	.944	**0.040**	1131.829± 1431.810
FTS_{1200}	.952	.950	**.999**	0.050	359.545 ± 410.606
RTS_1	.267	.256	.695	1.007	2833.718 ± 4203.049
RTS_2	.035	.033	.958	1.008	445.645 ±527.969

Table 2: Evaluation on London data. The first three columns show the measures adopted to test our new approach. The fourth one reports the ratio between the average sampling period of non-stop points over that of all points, and the last column is the number of trajectories.

There is also a minority of users having values outside the two peeks.

Comparison of Evaluation Measures

In this section we compare the proposed self-adaptive trajectory segmentation approach with the other methods taken into account. In Tables 1 and 2 we report the results obtained with all the methods. The first three columns show the evaluation measures described above. The fourth column shows the ratio between the average sampling period of movement points (thus discarding the stop portions of the user's trajectory) and the average sampling period of the full trajectory, while in the last one the average number of segments with its standard deviation is given. In general, we can observe that the best results were obtained with the ATS and FTS methods, both for Rome and London. Analyzing the ratio (fourth column) we can see that values are low for both ATS and the FTS ones, meaning that the long stops are ignored (i.e. they are recognized as real stops) and just the short ones are considered. On the contrary, with the random approaches the ratio is bigger because the algorithm function evaluates all stops in the same way. Looking at the number of segments it is possible to note that FTS and ATS methods produce different quantities, especially the FTS_{120} result produces less segments in the Rome case and much more in London. About the last two approaches, the RTS_1 method works with a random number of

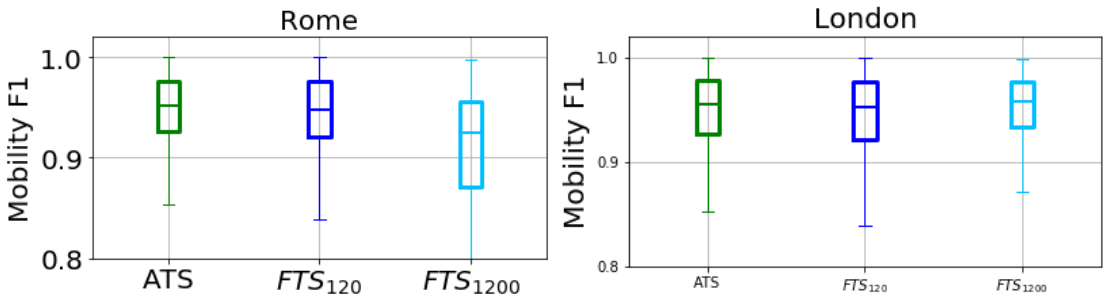

Figure 5: Boxplots for the $MF_{.25}$ results. On the Rome data ATS yields better results than the FTS solutions, while in London all three produce almost the same results. The variability of ATS results is consistently smaller than the other methods, which is a sign of robustness.

segments, so it is normal that the final result differs from the others, while the RTS_2 takes as number of segments the same of the ATS approach so we aspect to achieve similar results.

For the evaluation measures we can see that our new approach reached the goal we expected, i.e., yielding a quality of results which is always comparable or higher than fixed-threshold approaches and more robust. Indeed, for both Rome and London the values obtained by ATS are compatible with the FTS results, even better in the $MF_{.25}$ for Rome and in the distance coverage for London. In particular, in the Rome example, having a high $MF_{.25}$ values means that also the time precision and the distance coverage are well correlated in a way that produce a satisfying result. If we see the FTS_{120} result we can note that the time precision is high but the distance coverage is very low because the algorithm builds short trajectories with few points. An analogous reasoning can be done analyzing the FTS_{1200} method which produces an excellent distance coverage score but a lower time precision. Our solution reaches a good balance, thanks to its self-adaptive characteristic that allows to control and correct the trajectory fragmentation, and all its evaluation measures are always either the best or the second best of the group.

To have a better understanding of the quality of our new approach, the distribution of $MF_{.25}$ values for the different approaches on the two datasets is shown in Figure 5 through a boxplot visualization. For the Rome case we can observe that with the ATS approach the median value is the highest (closest to 1) and the inter-quartile range is smaller than the other two, meaning that we have a smaller variabiliy and thus more robust results. The London case appears to be different, and the best $MF_{.25}$ values are obtained with the FTS_{1200}, with a median similar to ATS and a slightly narrower box. This leaves room for future improvements of our methodology.

Comparison of Segmentation Statistics

In the following we analyze other statistical indicators on the trajectory segments extracted by the various methods. The next plots want to show other significant features for the segmentation problem in order to compare their distribution and try to infer something more about the segmentation. In addiction discovering some hidden correlations between trajectory features and

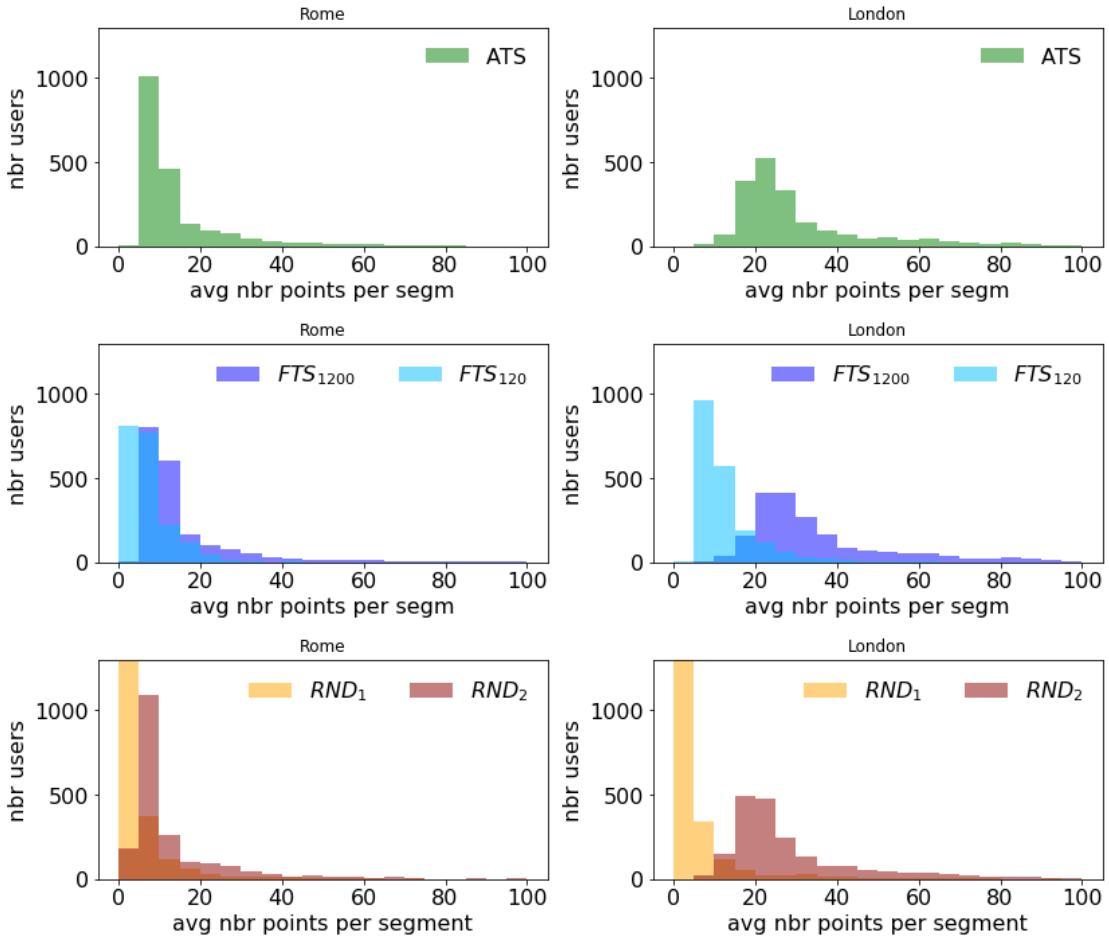

Figure 6: Distributions of average number of points per segment in Rome (left) and London (right).

the segmentation approach could lead to a better understanding of the problem and highlight other relevant aspects. In Figure 6 we report the distributions of the average number of points per segment for Rome and London. For all methods, the majority of segments have less than 20 points, probably meaning that most of the trips take place within the city. However, in the distribution tails some long trajectories with more points emerge. We observe that the distribution peaks of ATS place somehow in between the peaks of the two FTS variants (though closer to FTS_{1200}, especially in London) thus finding a trade-off between them. Moreover we can see that London and Rome distributions are different: London has a wider distribution than Rome, meaning that the variety of trips is greater in London.

In Figure 7 are displayed the distributions of the average number of segments per user. In London most of the users have less than 20 trajectory segments. The peak of the distribution is between 5 and 10 segments. Between 30 and 100 segments the distribution remains stable at a

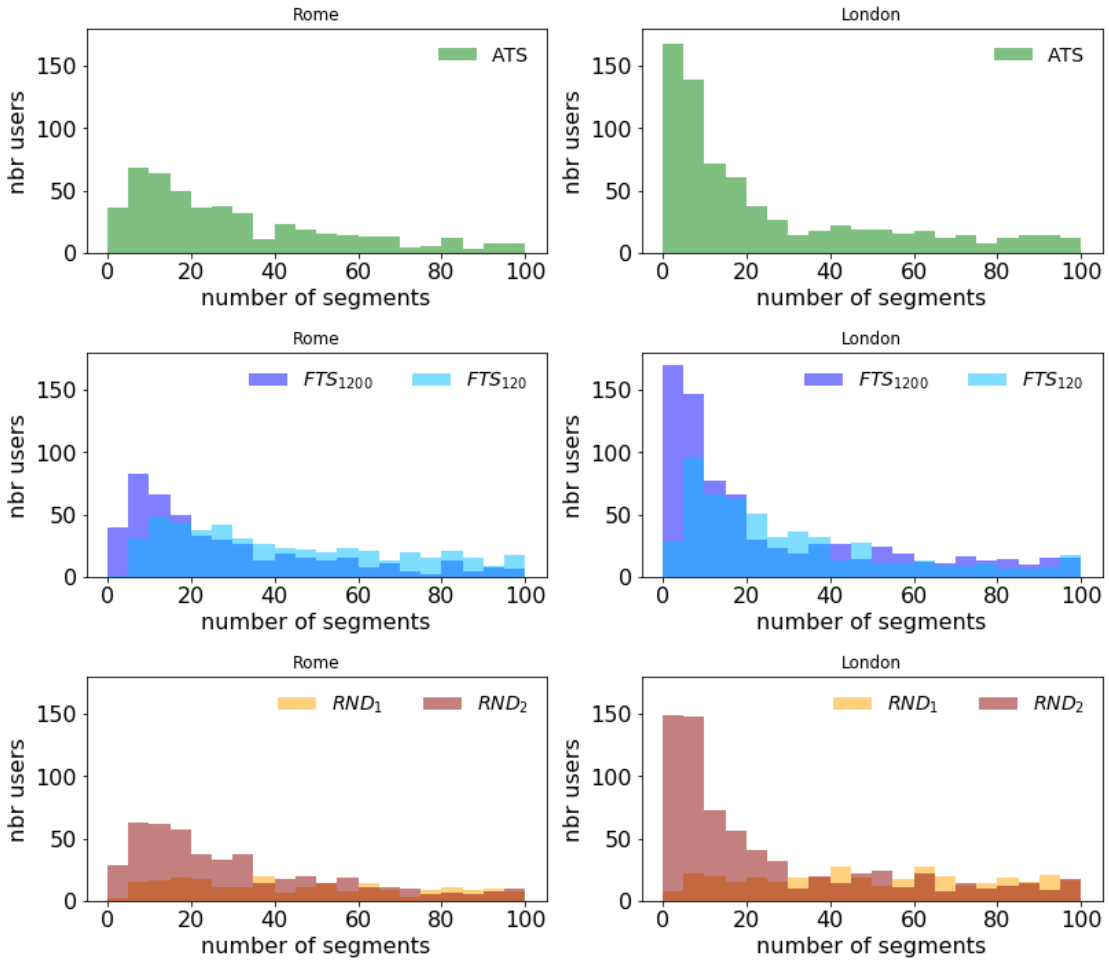

Figure 7: Distribution of the number of trajectory segments over Rome (left column) and London (right column) with each segmentation method (on the rows, grouped by family).

small value larger than zero. In Rome we observe a similar result with a peak between 15 and 20 trajectories. Also in this case, the peak of ATS distribution tends to stay in the middle of the FTS ones.

In Figure 8 we compare the distribution of average length and average duration of the segments returned by ATS (left) and FTS (right) for the area of Rome. With the ATS method the peak value is around 10km, thus confirming that most of the trips are short, and likely to take place around the city. With the FTS methods the peak position depends on the temporal threshold imposed: with a threshold of 1200 seconds the average distance is similar to ATS, while with 120 seconds it becomes lower and close to 5 km. The results for the RTS methods are omitted, since their plots are very similar to the FTS ones. Also, the plots in London show exactly the same kind of behaviour observed on Rome.

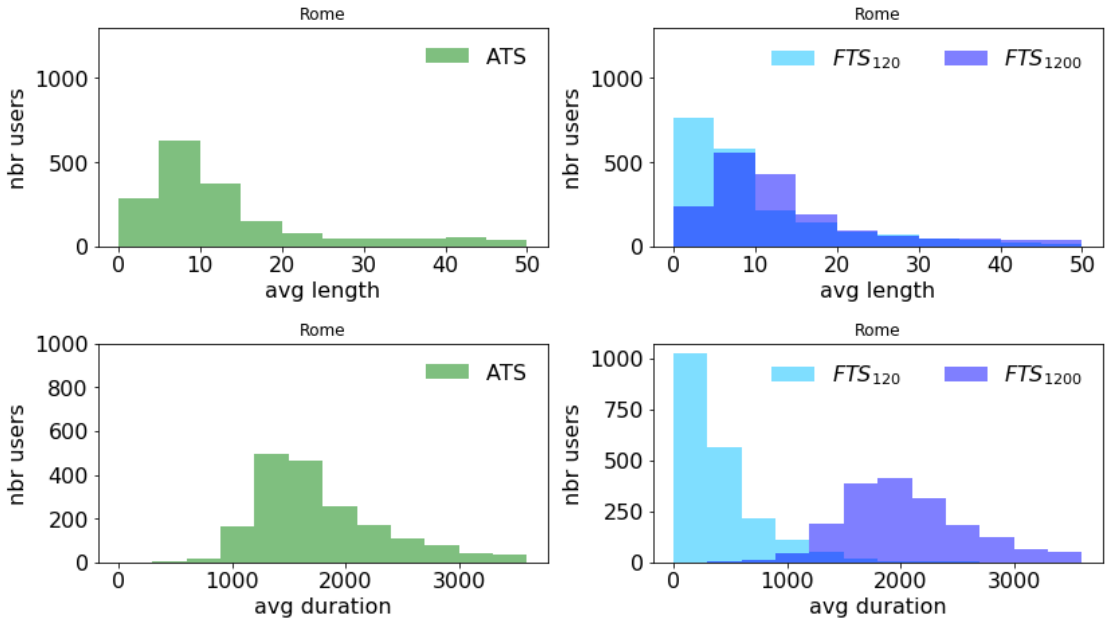

Figure 8: Distributions of the average length (top) and duration (bottom) for the trajectory segments returned by ATS (left) and FTS (right) for the area of Rome.

In terms of segment duration, ATS yields a distribution with a peak around $1200 - 1500$ seconds ($\sim 20 - 25$ minutes). With the FTS methods the peaks change: for FTS_{120} the peak is around 500 seconds while for FTS_{1200} the peak is centered in 1800 seconds. Also in this case, the results on London are very similar and omitted here.

Case Study

In this section we show qualitatively on a case study the effectiveness of ATS with respect to FTS. In Figure 9 we report the segmentation returned by FTS_{1200} [179] (left) and by ATS (right), the user is travelling from south to north. FTS_{1200} [179] returns two trajectories (green and blue), while ATS returns three trajectories (green, orange and blue). The second line of plots report the inter-leaving time between consecutive GPS points. The colors match the trajectory segments, while stops are highlighted in red. We observe how ATS identifies the short stop of less than 15 minutes at the service area similarly to the subsequent longer stop. On the other hand, FTS_{1200} considers the first stop as part of the green trajectory. The map in the bottom line of Figure 9 shows the service area which is very close to the GPS points reported on the bottom right corner of the map. This case study highlights how various existing stops under a certain predefined threshold can be missed with a segmentation approach like FTS, while a more data-driven and self-adaptive method like ATS is able to take into account specific user behavior and return a better result.

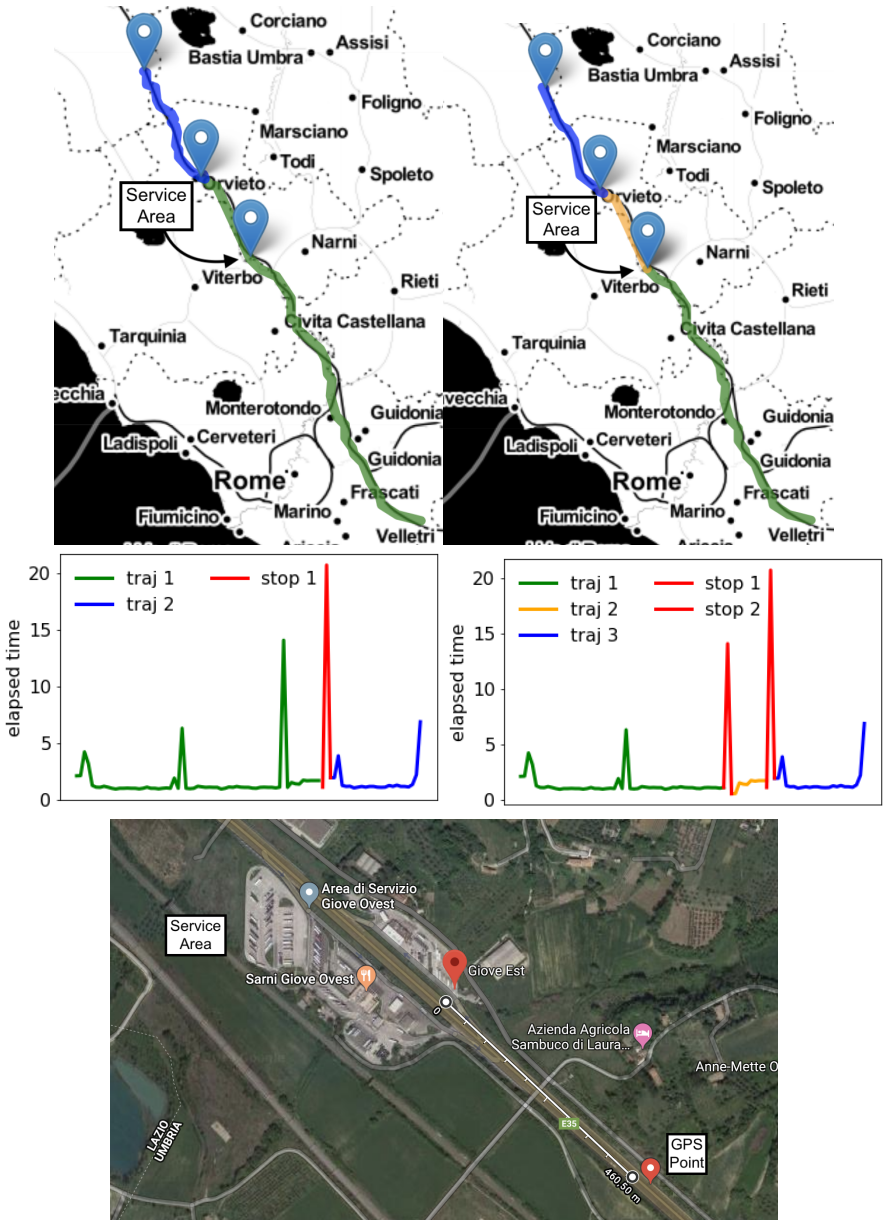

Figure 9: Trajectory segmentation returned by FTS_{1200} (left) and ATS (right). The user is travelling from South to North. *Top*: spatial representation showing the trajectory segments. *Center*: temporal segmentation showing the inter-leaving time between GPS points. *Bottom*: zoom on the service area highlighted in the top maps where the user probably stops for $\sim$ 15 minutes. Best view in color.

3.1.7 Adaptive location extraction with TOSCA

Based on the set of points where a user stopped, our aim is to identify subsets that most likely represent the same place. Following [66], we formally define the *Locations Detection Problem*

(LDP) in three steps:

Definition 5 (Location set) *Given a set of stop points P, a* location set $\mathcal{L}$ *for P is a partitioning of P into disjoint sets: $\forall l \in \mathcal{L} : l \subset P$, $\bigcup_{l \in \mathcal{L}} l = P$ and $l, l' \in \mathcal{L} \wedge l \neq l' \Rightarrow l \cap l' = \emptyset$.*

In general, the locations can be either provided as input, to be considered as ground truth, or they can be inferred (detected) directly from the stop observations through algorithms.

Definition 6 (Real and Detected Locations) *Given a set of observations P we denote the* real locations *associated to P as $\mathcal{L}_P = \{L_1, L_2, \dots L_k\}$, and the locations inferred (detected) from P through any algorithm with $\mathcal{D}_P = \{D_1, D_2, \dots D_k\}$.*

The Locations Detection Problem, then, is simply defined as the task of inferring locations as close as possible to the real ones, across all the users:

Definition 7 (Locations Detection Problem) *Given a set of users U and their observations $\mathcal{P}_U = \{P_u\}_{u \in U}$, the* Locations Detection Problem *(*LDP *in short) consists in producing for each set of observations $P \in \mathcal{P}_U$ a partition $\mathcal{D}_P$ that is similar to the corresponding real partition $\mathcal{L}_P$.*

We consider the most common case, where no real locations are known a priori, and therefore the problem requires to perform an unsupervised learning. In particular, it can be seen as a partitive clustering task, and is generally solved in literature through the adoption of a clustering algorithm. The result is a set of clusters of observations, which correspond to the detected locations, i.e. $\mathcal{D}_P$.

TOSCA

The TOSCA approach can be summarized as the combination of two main steps:

1. extract (several small) clusters and corresponding medoids through center-based methods. In particular, the X-Means algorithm was selected through empirical evaluations;
2. cluster the medoids through a Single Linkage hierarchical algorithm. Stop the iterative clusters aggregation (or, equivalently, cut the *dendogram* resulting from a complete run of the algorithm) through a statistically-determined threshold on the increase of distance between the clusters to be merged at each iteration.

The input of the method are the set of stop observations P of a user and a *cut-criteria*. The first step of TOSCA consists in clustering P with X-Means, extracting the corresponding medoids M. *X-Means* [145] is a fast and statistically founded refined version of K-Means. Given an interval [k_{min},k_{max}] it finds the set of clusters which minimizes the *Bayesian Information Criterion (BIC)* by using smart centroids and values of k. In the general case, the parameters can be simply set to k_{min}=2 and k_{max}=|P|-1, while smaller intervals can be used if knowledge about k is available, to reduce the search space and speed-up the computation.

The second step executes a Single Linkage clustering on the set M of medoids. *Single Linkage* [163] is a standard agglomerative hierarchical clustering methods that builds a hierarchy of clusters by progressively joining the two closest elements at each step. The resulting hierarchy is called *dendrogram*, and it shows the sequence of cluster fusions and the distance at which each fusion took place. The final clustering is generated by cutting the dendogram $\mathfrak{D}$ at distance (height) *dist* according to the *cut-criteria*. The dendogram can be mathematically represented by a list $\mathfrak{D} = [d_0, d_1 \dots d_{|M|-1}]$ of the distances computed by Single Linkage to aggregate the clusters, i.e. d_i is the distance at which two clusters are aggregated at iteration i.

The *cut-criteria* considered in the algorithm comes from the outlier detection theory. The idea behind this choice is the fact that empirical experiments show how the differences between consecutive distances in $\mathfrak{D}$ contain sudden spikes indicating the change of trend in the aggregations of the clusters. Among various alternatives for identifying such spikes, the *Thompson Tau Test* was chosen, which takes into account the mean μ and standard deviation σ of a distribution, and provides a statistically determined rejection region.

3.1.8 Impact on location extraction

The combined effect of the flexible trajectory segmentation method and the successive location extraction is expected to yield a different set of locations w.r.t. previous solutions based on fixed thresholds. In particular, taking as reference the typical fixed threshold of 1200 seconds (FTS_{1200}), Figure 4 shows that the flexible solution finds similar values in $\sim 40\%$ of cases, and smaller ones in the others. That suggests that the new method might find more significant stops and therefore identify more locations. Figure 10 shows the results of a direct comparison of the number of locations found by the two approaches, expressed as relative increase of locations gained by our method.

The plot summarizes the results obtained on 1176 vehicles belonging to Pilot 1 dataset, having a variable number of observed points that ranges from 4429 to 11061. As we can see from the figure, there is a large number of cases where the differences are relatively small, in smaller quantities also negative, and then a group where the novel method adds a large amount of locations.

Overall, the average relative increase of our method is 0.183621 (i.e., 18.36%), meaning that the very significant impact on the trajectory segmentation we saw in previous sections does not completely translate into a large variation on the locations found. While a definite explanation for that is still under investigation, the current hypothesis is that two contrasting effects cancel out mutually: on one side, more stops are found, several of which represent locations visited occasionally that would be otherwise lost; on the other hand, sparsely covered locations sometimes are mistaken by the algorithms into two or more separate ones, and having more observations for that (real) location results into merging the different pieces, thus reducing the final number of locations.

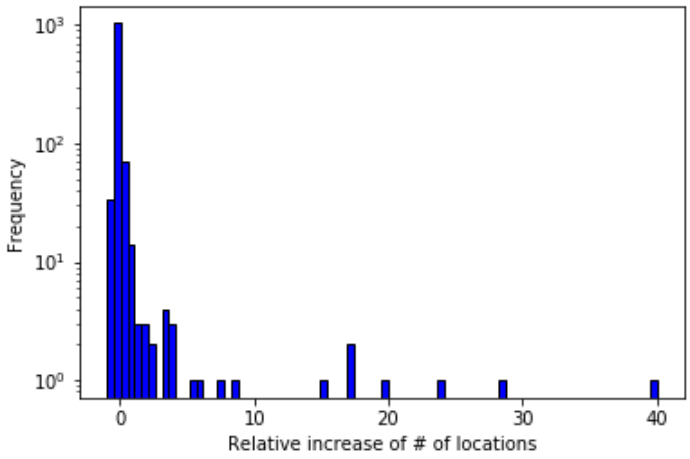

Figure 10: Relative increase of number of locations extracted w.r.t. FTS_{1200}.

3.1.9 Conclusion

The algorithms presented in this section form a user adaptive method for solving the location extraction problem, a very common and useful task in mobility data mining. The experiments show that it is possible to derive user-adaptive cut thresholds for the trajectory segmentation step, improving the performances over less flexible solutions. Also, the more adaptive extraction of stops allows to improve the location extraction process, in some cases significantly increasing the number of relevant locations discovered in users that have lower personal stop durations, which would be otherwise lost by other methods.

While the results are satisfying and the tool showed to be of practical utility, especially in the Track&Know context, several improvements are currently being explored. Among them, we mention a possible trajectory segmentation refinement which is not only user-adaptive, but also location-adaptive, thus considering the fact that a stop at different places might require time intervals of different duration to be considered a significant stay – and thus a trajectory cut point. Also, we are exploring the idea of using the context around the (moving) user to improve results, for instance looking at the mobility of other users and the geographical area surrounding the candidate stops.

Finally, we remark that this component provides a fundamental tool for analyses aiming to model the mobility of single individuals. That is in particular the case of most of the tools developed in Task 4.2 of the project, and reported in deliverable D4.2: beside using the location extraction tool for building Individual Mobility Networks (the core component of the methods in Task 4.2) the deliverable provides a data flow schema and a platform integration and testing section that also includes trajectory segmentation and location extraction as atomic components.

3.2 Analysis of electrificability of trips

In summary:

- *Generic question addressed*: Measure electrificability of a trip.
- *Track&Know specific question*: Measure the battery consumption of a route and of a daily sequence of trips.
- *Novelty / Advantage over existing methods*: Exploits detailed information of real GPS trajectories.
- *Experiments conducted*: Testing on a Track&Know Pilot dataset with quantitative evaluations.
- *Type of analytics*: Descriptive analytics.
- *Automation / TRL*: TRL 4 (automated tool, tested on real data for simulation-based applications)
- *Extension to other domains*: None; the tool is specific for vehicle mobility data.

Electric mobility is one of the main advocated solutions for making urban environments ecologically more sustainable, improving the quality of life of citizens. However, most users are very little familiar with what driving an electric vehicle (EV) really means and what it might change in their daily life if they replace their fuel-based vehicle with and electric one. This lack of knowledge causes several worries to the average potential user, even though its many advantages for the environment are clear. One of the biggest differences between a fuel-powered vehicle and a battery-powered vehicle lies in the reduced autonomy: while high-profile EV models have performances similar to fossil-fuel cars, average EVs have a range in the order of 200 km, which makes the need for recharging more frequent. Also, the time required to fill a fuel tank is usually less than a quarter of an hour, while a stop to recharge the battery of an electric vehicle can easily take more time, up to some hours, depending on the capacity of the battery and the type of recharger. Finally, at the present the recharge infrastructures are much less developed than fuel ones, thus arising further concerns about the capability for a user to satisfy their mobility needs.

In this work we developed a tool that provides basic functionalities for addressing some of the concerns mentioned above. In particular, we aim to evaluate whether the existing mobility demand of a user could in principle remain sustainable with an EV, i.e. whether their trips could be performed exactly as they are (same origin and destination, same route, same speed) without running into battery shortage. We realize this by analyzing the recorded trajectories that a user performed with a fuel-based vehicle and estimating the battery consumption (in

KWh) for each trip in input. Since the output is mainly estimating consumptions, it can be seen as an enrichment process that assign a measure to each trip. In this section we also make use of this basic tool to simulate the whole history of the user, thus considering how the consumption of each trip adds up to the previous ones, assuming that the vehicle can recharge the battery whenever it stops for a significant amount of time either at home or at a recharge facility.

The approach described in this section is also linked to tools developed within Task 4.2 and having similar objectives, which are presented in deliverable D4.2. While the methods presented here mainly focus on simulating the single trip as it was performed by the user, the other solution performs a more complex analysis of the user mobility as a whole, and is based on a mobility data representation named Individual Mobility Network, also considering a wider range of scenarios.

3.2.1 Input movement data

The methodology assumes to have as input the traces of vehicles equipped with GPS devices, which record the GPS position and other related information, referred to as *observations*. Each observation typically contains the following information, which also match very closely the datasets used in the Track&Know pilots:

- id: anonymous vehicle identifier
- ts: timestamp, composed of date and time
- lat, lon: the coordinates of the position, expressed as latitude and longitude
- speed: the instantaneous speed of the vehicle
- heading: the direction of the car (not used in this work)
- quality: quality of position estimate, based on the number of satellites connected
- status: the status of the car engine (just switched on, switched off, running)
- delta: spatial distance travelled from the last observation

For practical reasons, like bandwidth limitation of communication costs, the on-board devices record the observations and send them to the data collector in a non-continuous way, by applying some spatio-temporal filters to reduce the amount of data transferred, meaning that not all the points of the trip are recorded. For this reason the stream might contain low sampling rate data (e.g. between 10 to 60 seconds between each observation, in some cases even longer).

3.2.2 Preprocessing and elevation enrichment

The initial stream of data can contain observation with poor localization accuracy, marked by appropriate values in the *quality* attribute. Moreover, the successive steps of our process require to know the altitude associated to the recorded positions, which is an infomation missing

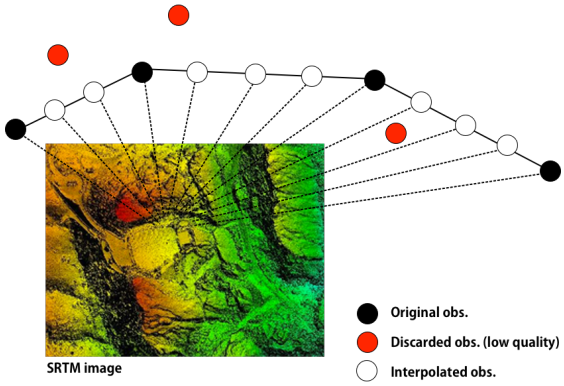

Figure 11: The reconstruction and altitude enrichment process

from most GPS tracking devices. Therefore, the following steps are performed, before the real computation:

- As first step, our tool excludes all the positions having low quality connection with the satellite, e.g. in experiments with Pilot 1 data, it was decided to keep only points with *quality=3*, which is the highest, and exclude all lower values.
- Then, a simple trajectory reconstruction is performed, which groups observations in single trips of the users according to the status information and spatial and temporal constrains – indeed, the status field alone is not always reliable. Also, in order to obtain a more accurate simulation the trajectory is also "completed" using a simple interpolation to guarantee a temporal granularity less then 2 seconds between two consecutive points.
- After the cut of the traces into trajectories the module extracts the elevation of the points from official cartography data using satellite images. This is a very important and time consuming task because the simple geo-localization is not enough for the simulation of the vehicle cars, due the fact that the degree of inclination of the road strongly affects the energy consumption, as better discussed in the next sections. To calculate this information, the data provided by the CGIAR Consortium for Spatial Information (CGIAR-CSI) website is used, which contains the measurements obtained by the Shuttle Radar Topography Mission (SRTM) for the Digital Elevation Data, representing the most accurate data source available to date. The data is stored as GeoTIFF, a multi-layered TIFF image that holds altitude information for each pixel. The module will request automatically the set of tiles (90m x 90m) covering the spatial area where the trajectories are located and will extract the information point-by-point, attaching it to the existing bi-dimensional points. The process is visually summarized in Fig.11.

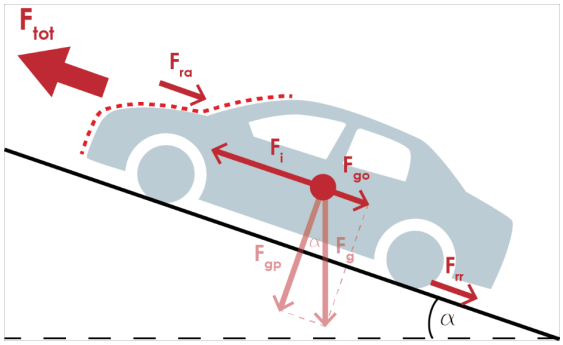

Figure 12: The physical forces to which a vehicle is exposed during the movement

3.2.3 Consumption estimation

Our estimation of battery consumption for each trip of the user is based on a instantaneous consumption model introduced in [184] and recommended in [8] as a good trade-off between realistic simulation and efficient computability. The model considers all the physical forces to which the vehicle is constantly exposed in order to estimate the amount of electric power needed to reach a certain speed, which are illustrated in figure 12.

The model is general and can be adapted to each type of vehicle changing specific parameters. In details it considers:

- Rolling resistance: $F_{rr} = R(M_{car} + M_d)gcos\alpha$. Where $R()$ Is the rolling resistance coefficient of the tire, M_{car}[kg] is the mass of the vehicle, M_d[kg] is the mass of the driver, g= 9.81 [m / s2] is the acceleration of gravity and $/alpha$ [rad] is the angle of the surface on which the machine is located with respect to the horizontal plane.
- Aerodynamic resistance: $F_a = \frac{1}{2}AC_d\rho v^2$. Where $A[m^2]$ is the area of the front surface of the vehicle, C_d is the aerodynamic drag coefficient, ρ= 1.2041 [kg / m^3] is the density of the air at 20°C and v[m/s] is the vehicle speed.
- Horizontal component of gravity: $F_{hc} = (M_{car} + M_d)gsin\alpha$
- Inertia: $F_{la} = 1.05(M_{car} + M_d)a$. Where $a[m/s^2]$ is the acceleration of the vehicle.

Hence, the total pulling force can be expressed as $F_{tot} = F_{rr} + F_a + F_{hc} + F_{la}$. Mechanical traction power is the product of the traction force and the average vehicle speed. It depends on the engine power and transmission efficiency and is: $P_{tot} = F_{tot}v$. This power is transferred to the wheels, and assuming an efficiency of the gears, the resulting power is:

$$P_{engineOut} = \frac{P_tot}{\eta_{gear}}$$

The efficiency of the engine is then considered and the power that enters is:

$$P_{engineIn} = \frac{P_{engineOut}}{\eta_{engine}}$$

The auxiliary power P_{aux}, which represents the power necessary for lights, air conditioning, radio, and other electrical appliances in the vehicle is considered as: $P_{consMov} = P_{engineIn} + P_{aux}$.

During breaking, cars must remove its kinetic energy converting it into heat. Electric Vehicles have regenerative breaking that can recover a fraction $R_{genRatio}$ of such energy to recharge the battery. In this case the mechanical power is negative, $P_{te}<0$, and this variable will be weighted by the regeneration factor and flow back from the wheels to the motor (working in generation mode) and to the battery: $P_{teReg} = R_{genRatio}P_{te}$.

The next step is taking into account the transmission efficiency. The power going out from this block to the electric machine-power electronics block is given by: $P_{regEngineOut} = \eta_{gear}P_{teReg}$. The power going out from the electric machine-power electronics block to the battery is given by $P_{regEngineIn} = \eta_{engine}P_{regEngineOut}$, and the final amount of energy is thus: $P_{finalBattery} = P_{consMov} + PregEngineIn$.

Since detailed information about the type of vehicles to simulate are usually not available, by default a medium class car model is considered. The vehicle parameters are the following:

- Cross section area 2.27 m^2
- Machine weight 1521 kg
- Driver weight 90 kg
- Aerodynamic drag coefficient 0.29
- Rolling resistance coefficient 0.012
- Regeneration ratio 0.25 default / 0.35 in eco mode
- Battery capacity 24kW
- Low battery limit 8-10%
- Transmission efficiency 0.95
- Machine and electric motor efficiency 0.98
- Battery charging efficiency 0.95
- Battery discharge efficiency 0.98

These settings can be easily changed by modifying a configuration file, where all parameters are listed.

The resulting model for charging and discharging [184] of the battery must consider the energy transmission efficiency coefficients: η_{charge} and $\eta_{discharge}$, thus the two operations produce energy as:

$$E_{bat} = \frac{P_{finalBattery}.\Delta t}{\eta_{discharge}}$$

$$E_{bat} = \eta_{charge} P_{finalBattery}.\Delta t$$

Therefore the variation in capacity will be:

$$\Delta Soc = \frac{E_{bat}}{E_{bat}^{CAP}}$$

where E_{bat}^{CAP} is the maximum capacity of the battery. Considering the variation as function of the time t the following model is obtained:

$$SoC[t] = SoC[t-1] - \Delta Soc - \delta_{self_disch}$$

where δ_{self_disch} is the self-discharge losses.

3.2.4 Model Implementation and output format

The implemented module estimates the percentage of battery used during each trip and, consequently, it can be used to understand and estimate how much a car needs to be recharged at the end or during the day. We considered two ways to recharge the car: (i) the machine stops near a charging station or (ii) when it returns home. The charging stations are provided to the module as list of geo-locations (may be an empty list, in this case the module will consider only the second option).

Whenever a trajectory end (with a stop) the module will calculate the distance between the point where the vehicle stopped and the closes charging station, if the vehicle is located near one of these areas, it will consider the stop time as recharging time until the start of the next trip of the user. Notice that in this tool we do not consider changes of routes to intercept recharge stations in case of need. That would require trip planning capabilities that go beyond the objectives of this module, and are instead part of the whole-user simulations developed in Task 4.2 (see deliverable D4.2 for details).

In the case of the home charging we consider the time spent by the user in its home location (here simply computed as the place where he stops for the night) which usually leads to a full recharge of the battery.

The output of the tool is a table that reports useful statistics of each trip, including battery consumption of the single trip, and cumulative information within the day. A sample result is shown in Fig.13, where the attributes are the following:

- user id: the anonymous identifier of the vehicle;

	user_id integer	progressive_number integer	start_time character varying (1	start_lat double precisi	start_lon double preci	end_time character varying (20	end_lat double preci	end_lon double preci	residual_charge_before double precision	consume double pr	total_distance double precisi	residual_charge_after double precision	recharge_flag smallint	recharge_amount double precision	trajectory_overflow_flag smallint
1	1610	1	02/01/17 06:57:42	43.85472	10.557471	02/01/17 07:14:07	43.850272	10.514788	100	1.31	3.46	98.69	0	0	0
2	1610	2	02/01/17 12:02:34	43.85024	10.514826	02/01/17 12:13:18	43.85586	10.557006	98.69	1.53	3.58	97.16	0	0	0
3	1610	3	02/01/17 13:02:09	43.855496	10.5575	02/01/17 13:12:25	43.850168	10.515004	97.16	1.72	3.46	95.44	0	0	0
4	1610	4	02/01/17 18:11:13	43.850112	10.514889	02/01/17 18:21:57	43.855808	10.557078	95.44	1.94	3.53	93.5	1	100	0
5	1610	5	03/01/17 07:07:59	43.855772	10.557785	03/01/17 07:18:17	43.850256	10.514844	100	1.67	3.5	98.33	0	0	0
6	1610	6	03/01/17 12:16:48	43.850828	10.514071	03/01/17 12:28:19	43.855836	10.556939	98.33	1.62	3.57	96.71	0	0	0
7	1610	7	03/01/17 12:59:28	43.855836	10.556941	03/01/17 13:10:21	43.85022	10.514902	96.71	1.75	3.44	94.97	0	0	0
8	1610	8	03/01/17 17:57:05	43.850252	10.514915	03/01/17 18:08:56	43.85584	10.557088	94.97	1.78	3.54	93.19	0	0	0
9	1610	9	03/01/17 19:50:46	43.855816	10.557172	03/01/17 20:00:59	43.849944	10.573134	93.19	0.85	2.29	92.34	0	0	0
10	1610	10	03/01/17 21:51:07	43.84092	10.573284	03/01/17 21:59:13	43.855804	10.557034	92.34	1.27	2.76	91.06	1	100	0
11	1610	11	04/01/17 07:18:14	43.855844	10.557199	04/01/17 07:44:50	43.83666	10.50781	100	2.46	6.7	97.54	0	0	0
12	1610	12	04/01/17 14:01:52	43.836952	10.507938	04/01/17 14:45:34	43.851248	10.511672	97.54	4.29	10.49	93.24	0	0	0
13	1610	13	04/01/17 18:25:24	43.851312	10.511637	04/01/17 18:36:28	43.855756	10.557092	93.24	1.78	3.71	91.46	1	100	0
14	1610	14	05/01/17 07:01:35	43.855792	10.557294	05/01/17 07:13:15	43.850304	10.514822	100	1.68	3.49	98.32	1	100	0
15	1610	15	05/01/17 18:06:43	43.850632	10.514812	05/01/17 18:19:54	43.855796	10.557037	100	1.54	3.55	98.46	1	100	0
16	1610	16	06/01/17 13:52:22	43.855692	10.557036	06/01/17 14:06:06	43.838344	10.507442	100	3.4	4.79	96.6	0	0	0
17	1610	17	06/01/17 16:42:55	43.838352	10.507232	06/01/17 16:55:34	43.855784	10.55708	96.6	2.25	4.55	94.35	1	100	0

Figure 13: The resulting enriched trajectories with the estimation of the battery consumption.

- progressive number: progressive number of the trip (reset every day);
- start time, start lat, start lon: time and location where the trajectory started;
- end time, end lat, end long: time and location where the trajectory ended;
- residual charge before: status of the battery before the trajectory started;
- consume: consumption of the battery for the trajectory;
- total distance: the trajectory length;
- residual charge after: status of the battery after the trajectory;
- recharge flag: the flag is different from 0 if the battery is recharged at the end of the trajectory (1 at home, 2 at a charge station).
- recharge amount: considering the stop time, the amount of battery recharged;
- trajectory overflow flag: this flag is 0 if the trajectory is feasible with the battery and the consumption simulated (i.e. the vehicles does not run out of charge), 1 if not.

3.2.5 Tests and case study

The dataset used to test the tool belongs to Pilot 1 of the Track&Know project, and contains 21.577.813 observations that, after the preprocessing, form 2.341.162 trajectories belonging to 2.294 distinct vehicles. The trajectories are located in Tuscany (Italy) in a period of 3 months. In the following we first present qualitative results of the simulation (our case study), and then provide scalability tests of the tool.

The result of the computation with only the home recharge is shown in Fig.14, where only the 5% of the trips run out of battery considering the sequence of trips and cumulating the daily battery consumption, meaning that the mobility is more than feasible in this area with electric cars; on the other hand, it is clear that this 5% is distributed across the majority of the

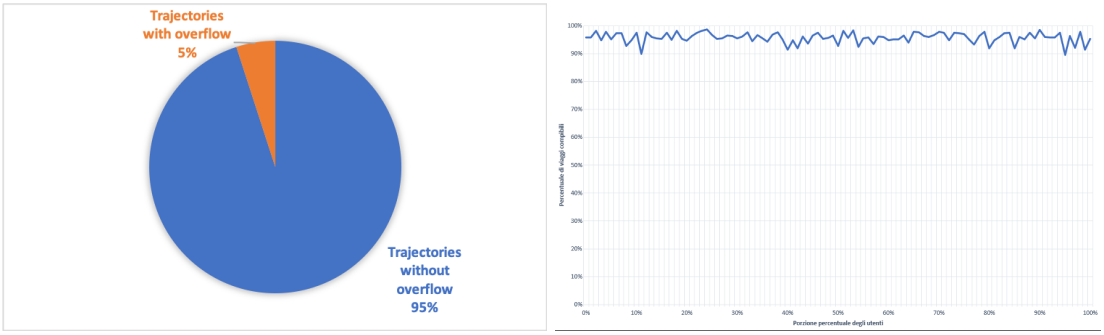

Figure 14: Percentage of simulated EV trajectories in battery overflow (left) and its distribution across users (right).

users, meaning that the recharge infrastructures (charge stations) need to complement in-house recharges in order to make EVs a complete replacement of fuel-based vehicles.

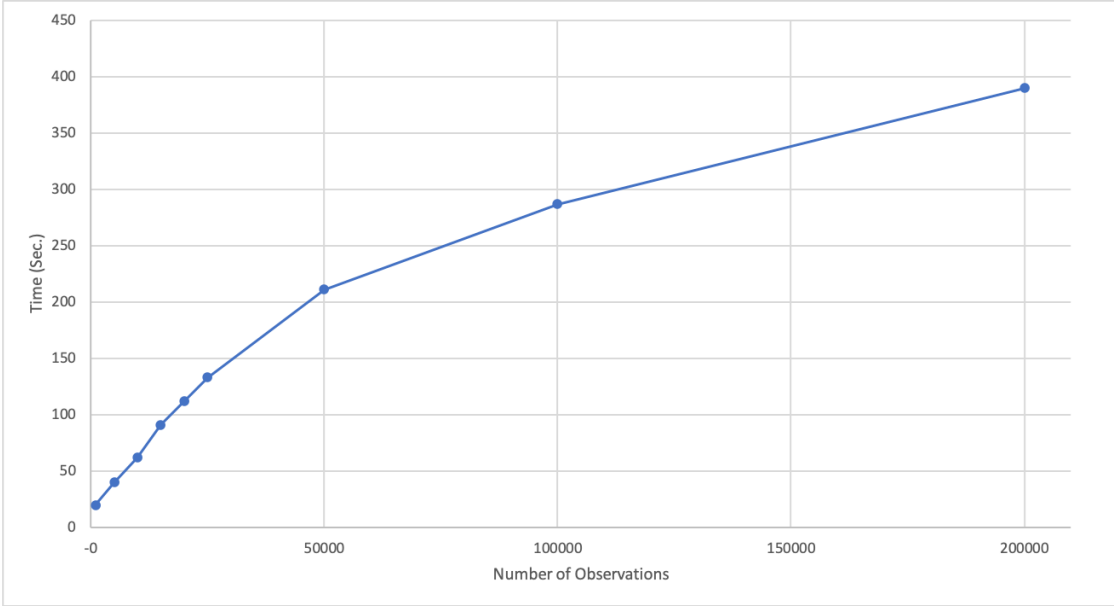

Figure 15: Computation time of EV trajectory simulation over different dataset sizes

Scalability tests

To test the performances of the module we performed several experiments increasing the size of original observations, with results reported in Figure 15. It is worth to consider the fact that the main bottleneck in the process is the query to the GeoTIFF data, consisting in a picture in high definition (6001 x 6001 pixels each tile of 90m x 90m). Moreover due the size of the tiles it is impossible to load in memory all of them so the module will keep in memory a buffer of

them during the process. For this reasons, the computation time grows less than linearly, since a larger number of trip simulations means a better usage of buffered GeoTIFF tiles, which dominate the cost for small size tests. The cut-off point in our setting was around 100000 observations, after which the cost becomes linear, with an approximate average execution time of $0.001s$ per observation and $0.0092s$ per trajectory.

Tool usage and integration

The method basically analyzes each day of each user separately, therefore a high parallelism of execution can be implemented. In case of streaming data the tool can be used in combination of a dispatcher separating the data into partitions, for instance by user, and have each partition processed by a separate instance of the tool (worker) without need to communicate with the others, as shown in Fig.16.

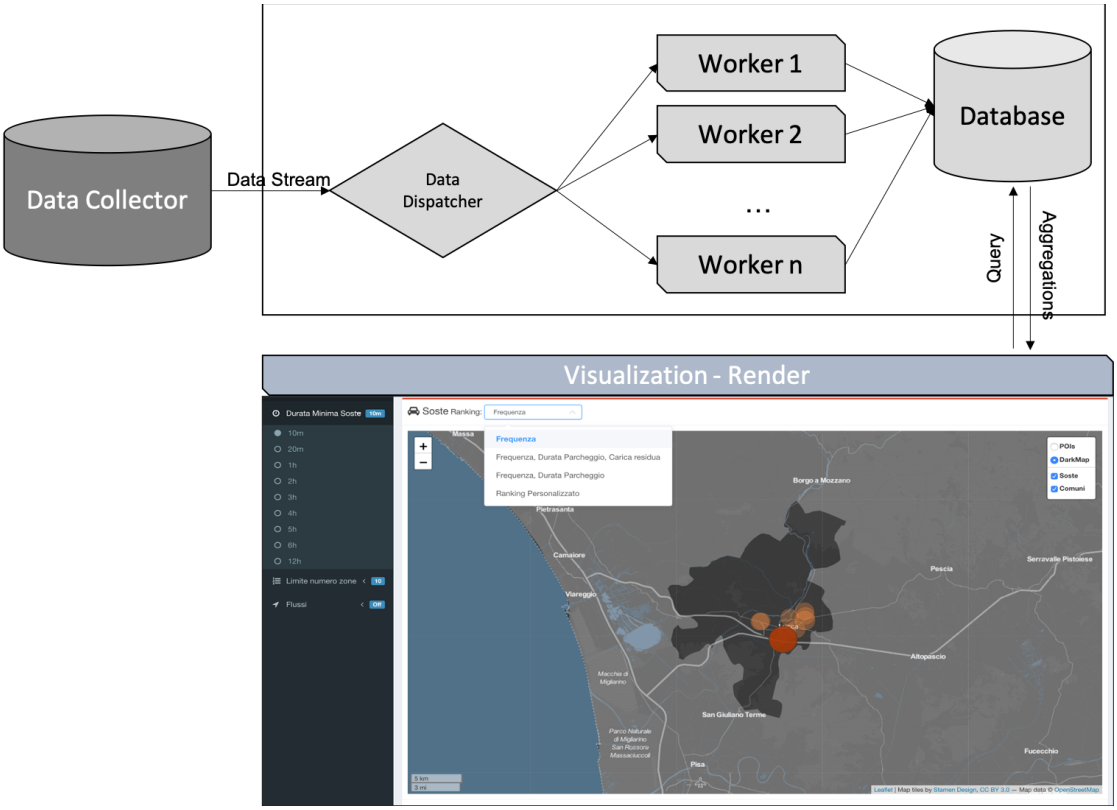

Figure 16: Usage and integration schema of electrificability simulation

The tool can be used as an enrichment step of trajectory data, therefore successive analyses can be easily accommodated. For instance, Figure 16 shows its application in a visual platform able to display the distribution of the stops, e.g. to evaluate possible good candidate locations for new charge stations. The process can be also iterative allowing the analyst to mod-

ify/adding/removing the stations and analyzing the changes, which simply requires to update the file containing the list of known recharge stations and their location.

3.2.6 Conclusion

This section introduced a method to simulate the battery consumption of an EV that moves by perfectly following the real trajectories of a user, exploiting a state-of-art consumption model. Performance results show a quasi-linear cost, especially for medium and large simulations, making it applicable to big datasets. Also, a preliminary case study suggests that the rate of daily trips that could be sustainable with a basic EV model is very high, thus confirming the viability of EV as a replacement of standard cars. Finally, an alternative, more sophisticated simulation approach is developed in deliverable D4.2 "Analytics for individuals' mobility networks", which exploits the same consumption model while building the simulation on top of an Individual Mobility Network model, allowing a more complete analysis of the user and the study of various scenarios.

3.3 Distributed Sub-trajectory Clustering

In summary:

- *Generic question addressed*: Discover clusters of sub-trajectories.
- *Track&Know specific question*: Identifying clusters of moving objects can have several applications, such as fleet optimization and predictive analytics.
- *Novelty / Advantage over existing methods*: Previous methods mainly focus on clustering entire trajectories, which might lead to loss of patterns that might exist for shorter periods, and also our parallel algorithms are designed and implemented in MapReduce so our solution scale gracefully for Big Data.
- *Experiments conducted*: Quality and performance/scalability tested against a 27GB dataset from SIS.
- *Type of analytics*: Descriptive analytics
- *Automation / TRL*: TRL level 3 (proof-of-concept implemented and tested, so we are between TRL 3 and 4)
- *Extension to other domains*: The method is applicable for other types of mobility data, e.g., maritime and aviation.

Trajectory clustering is an important operation of knowledge discovery from mobility data. Especially nowadays, the need for performing advanced analytic operations over massively produced data, such as mobility traces, in efficient and scalable ways is imperative. However, discovering clusters of complete trajectories can overlook significant patterns that exist only for a small portion of their lifespan.

3.3.1 Introduction

The unprecedented rate of trajectory data generation, due to the proliferation of GPS-enabled devices, poses new challenges in terms of storing, querying, analyzing and extracting knowledge from big mobility data. One of these challenges is cluster analysis, which aims at identifying clusters of moving objects (thus, unveil hidden patterns of collective behavior), as well as detecting moving objects that demonstrate abnormal behaviour and can be considered as outliers.

The research so far has focused mainly in methods that aim to identify specific collective behavior patterns among moving objects, such as [95, 90, 83, 130, 104, 103, 173, 201, 43]. However, this kind of approaches operate at specific predefined temporal "snapshots" of the dataset, thus ignoring the route of each moving object between these sampled points. Another line of research, tries to identify patterns that are valid for the entire lifespan of the moving objects [123, 141, 34, 162]. However, discovering clusters of complete trajectories can overlook significant patterns that might exist only for some portions of their lifespan. The following motivating example shows the merits of subtrajectory clustering.

Example 1 (Subtrajectory clustering) *Figure 17(a) illustrates six trajectories moving in the xy-plane, where each one of them has a different origin-destination pair. More specifically, these pairs are* $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $B \rightarrow A$, $B \rightarrow C$ *and* $B \rightarrow D$. *These six trajectories have the same starting time and similar speed. A typical trajectory clustering technique would fail to identify any clusters. However, the goal of a subtrajectory clustering method is to identify 4 clusters (*$A \rightarrow O$ *(red),* $B \rightarrow O$ *(blue),* $O \rightarrow C$ *(purple),* $O \rightarrow D$ *(orange)) and 2 outliers (*$O \rightarrow A$ *and* $O \rightarrow B$ *(black)), as depicted in Figures 17(b).*

The problem of subtrajectory clustering is shown to be NP-Hard (cf. [2]). In addition, the objects to be clustered are not known beforehand (as in entire-trajectory – from now on – clustering algorithms), but have to be identified through a trajectory segmentation procedure. Efforts that try to deal with this problem in a centralized way do exist [98, 144, 2], however, applying these centralized algorithms over massive data in a scalable way is far from straightforward. This calls for parallel and distributed algorithms that address the scalability requirements. In this context, one challenge is how to partition the data in such a way so that each node can perform its computation independently, thus minimizing the communication cost between nodes, which is a cost that can turn out to be a serious bottleneck. Another challenge, related to partitioning, is how to achieve load balancing, in order to balance the load fairly between the different nodes. Yet another challenge is to minimize the iterations of data processing, which are typically required

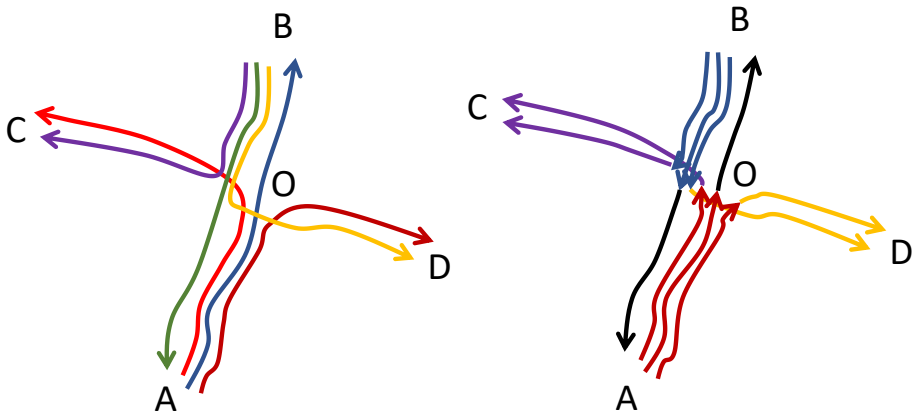

Figure 17: (a) Six trajectories moving in the xy-plane and (b) 4 clusters (red, blue, orange and purple) and 2 outliers (black).

in clustering algorithms. Interestingly, there have been some recent efforts towards mining mobility data in a distributed way, such as mining co-movement patterns [43], identifying frequent patterns [162] or adapting already existing distributed solutions to trajectory data [34], yet no approach for distributed subtrajectory clustering exists as of now.

Motivated by these limitations, we present the *Distributed Subtrajectory Clustering (*DSC*)* problem [168], which has not been addressed yet in a scalable and efficient way. Moreover, salient features of our approach include: (a) the discovery of clusters of subtrajectories, instead of whole trajectories, (b) spatio-temporal clustering, instead of spatial only, and (c) support of trajectories with variable sampling rate, length and with temporal displacement.

3.3.2 Related Work

In recent years, an increased research interest has been observed in knowledge discovery out of mobility data. Towards this direction, several *co-movement pattern discovery* and *trajectory clustering* methods, which are directly related to our work, have been proposed.

Co-movement patterns. One of the first approaches for identifying such collective mobility behavior is the so-called flock pattern [95, 185]. Inspired by this, a less "strict" definition of flocks was proposed in [90] where the notion of a moving cluster was introduced. There are several related works that emerged from the above ideas, like the approaches of convoys [83, 130], swarms [104], platoons [103], traveling companion [173] and gathering pattern [201]. However, all of the aforementioned approaches are centralized and cannot scale to massive datasets. In this direction, the problem of efficient convoy discovery was studied both in centralized [130] and distributed environment by employing the MapReduce programming model [129]. An approach that defines a new generalized mobility pattern is presented in [43]. In more detail, the general co-movement pattern (GCMP), is proposed, which models various co-movement patterns in a unified way and is deployed on a modern distributed platform (i.e., Apache Spark) to tackle

the scalability issue. Even though all of these approaches provide explicit definitions of several mined patterns, their main limitation is that they search for specific collective behaviors, defined by respective parameters. Nevertheless, none of the above techniques tackles the subtrajectory clustering problem.

Trajectory clustering. Most of the aforementioned approaches operate at specific predefined temporal "snapshots" of the dataset, thus ignoring the route of each moving object between these "snapshots". Another line of research, tries to discover groups of either entire or portions of trajectories considering their routes. A typical strategy in dealing with trajectory clustering is to transform trajectories to a multi-dimensional space and then apply well-known clustering algorithms such as OPTICS [10] and DBSCAN [40]. Alternatively, another approach is to define an appropriate similarity function and embed it to an extensible clustering algorithm. In this direction, there are several approaches whose goal is to group whole trajectories, including T-OPTICS [123], that incorporates a trajectory similarity function into the OPTICS [10] algorithm. CenTR-I-FCM [141], a variant of Fuzzy C-means, proposes a specialized similarity function that aims to tackle the inherent uncertainty of trajectory data. Nevertheless, trajectory clustering is a computationally intensive operation and centralized solutions cannot scale to massive datasets. In this context, [34] introduces a scalable GPU-based trajectory clustering approach which is based on OPTICS [10]. Moreover, [162] attempts to identify frequent movement patterns from the trajectories of moving objects. More specifically, they propose a MapReduce approach by employing quadtree-based hierarchical grid in order to discover complex patterns of different granularity.

Subtrajectory clustering. Nonetheless, discovering clusters of complete trajectories can overlook significant patterns that might exist only for portions of their lifespan. To deal with this, another line of research has emerged, that of *Subtrajectory Clustering*. The predominant approach here is TraClus [98], a partition-and-group framework for clustering 2D moving objects (i.e. the time dimension is ignored) that enables the discovery of common subtrajectories. The algorithm first partitions trajectories to directed segments (i.e., subtrajectories) whenever the shape of a trajectory changes significantly, by employing the minimum description length (MDL) principle. Subsequently, the resulting subtrajectories are clustered by employing a modified version of the DBSCAN algorithm, which is applicable to directed segments. Finally, for each identified cluster the algorithm calculates a "fictional" representative trajectory that best describes the corresponding cluster. A more recent approach to the problem of subtrajectory clustering, is S^2T-Clustering [144], where the goal is to partition trajectories into subtrajectories and then form groups of similar ones, while, at the same time, separate the ones that fit into no group, called outliers. It consists of two phases: a Neighborhood-aware Trajectory Segmentation (*NaTS*) phase and a Sampling, Clustering and Outlier (*SaCO*) detection phase. In *NaTS* the trajectories are split to subtrajectories by applying a voting and segmentation process that detects homogenized subtrajectories w.r.t. the density of their neighborhood. In *SaCO* the most representative subtrajectories are selected to serve as the seeds of the clusters, around which the clusters are

formed (also, the outliers are isolated). A slightly different approach is presented in QuT-Clustering [143] and [171], where the goal is, given a temporal period of interest W, to efficiently retrieve the clusters and outliers at subtrajectory level, that temporally intersect W. In order to achieve this, a hierarchical structure, called ReTraTree (for Representative Trajectory Tree) that effectively indexes a dataset for subtrajectory clustering purposes, is built and utilized. An alternative viewpoint to the problem of subtrajectory clustering is presented in [2], where the goal is to identify "common" portions between trajectories, w.r.t. some constraints and/or objectives, cluster these "common" subtrajectories and represent each cluster as a pathlet, which is a point sequence that is not necessarily a subsequence of an actual trajectory. A pathlet can be viewed as a portion of a path that is traversed by many trajectories. In order to solve this problem, the authors in [2] prove that this problem is NP-Hard and propose some approximation algorithms with theoretical guarantees, concerning the quality of the solution and the running time. Similarly, in [206] the goal is to identify corridors, which are frequent routes traversed by a significant number of moving objects. As already mentioned, all of the above subtrajectory clustering approaches are centralized and cannot scale to the size of today's trajectory data.

3.3.3 Problem Formulation

Given a set D of moving object trajectories, a trajectory $r \in D$ is a sequence of timestamped locations $\{r_1, \ldots, r_N\}$. Each $r_i = (loc_i, t_i)$ represents the i-th sampled point, $i \in 1, \ldots, N$ of trajectory r, where N denotes the length of r (i.e. the number of points it consists of). Moreover, loc_i denotes the spatial location (2D or 3D) and t_i the time coordinate of point r_i, respectively. A subtrajectory $r_{i,j}$ is a sub-sequence $\{r_i, \ldots, r_j\}$ of r which represents the movement of the object between t_i and t_j where $i < j$ and $i, j \in 1, \ldots, N$. Let $d_s(r_i, s_j)$ denote the spatial distance between two points $r_i \in r$, $s_j \in s$. In our case we adopted the Euclidean distance, however, other metric distance functions might be applied. Also, let $d_t(r_i, s_j)$ denote the temporal distance, defined as $|r_i.t - s_j.t|$. Furthermore, let Δt_r symbolize the duration of trajectory r (similarly for subtrajectories).

3.3.3.1 Similarity between (sub)trajectories Subtrajectory clustering relies on the use of a similarity function between subtrajectories. Although various similarity measures have been defined in the literature, our choice of similarity function is motivated by the following (desired) requirements:

Variable sampling rate and lack of alignment. We make the realistic assumption that the trajectories do not have a fixed sampling rate and that different trajectories might not report their position at the same timestamp.

Variable trajectory length. We also assume that different trajectories might have different length (i.e. number of samples). This specification excludes euclidean-based similarity measures which deal with trajectories of equal length.

Temporal displacement. A property that a desired similarity measure for (sub)trajectory clustering should hold, is to allow trajectories that have some temporal displacement to participate to the same cluster.

Symmetry. Given a pair of (sub)trajectories r and s, an appropriate similarity measure between r and s should have the property of symmetry (i.e. $Sim(r,s)$=$Sim(s,r)$).

Efficiency. The computation of the similarity should be efficient enough in order to be able to deal with massive volumes of data, without compromising the quality of the results.

In order to meet with the aforementioned specifications we utilize the Longest Common Subsequence (LCSS) for trajectories, as defined in [186]. However, other trajectory similarity functions, which meet with the specifications set, are also applicable. More specifically, the LCSS utilizes two parameters, the parameter ϵ_t indicating the temporal range wherein the method searches to match a specific point, and the ϵ_{sp} parameter which is a distance threshold to indicate whether two points match or not. Hence, the similarity between two (sub)trajectories r and s is defined as:

$$Sim(r,s) = \frac{LCSS_{\epsilon_t,\epsilon_{sp}}(r,s)}{min(|r|,|s|)} \tag{1}$$

where $|r|$ ($|s|$) is the length of r (s respectively). Moreover, it holds that $Sim(r,s) = Sim(s,r)$.

However, LCSS returns the length of the longest common subsequence, which means that for a given point $r_i \in r$ that is matched with a specific point $s_j \in s$ the LCSS will consider the similarity between r_i and s_j as 1, regardless of their actual distance $d_s(r_i,s_j)$, which could vary from 0 to ϵ_{sp}. Put differently, LCSS considers as equally similar all the points that exist within an ϵ_{sp} range from r, which is a fact that might compromise the quality of the clustering results. Ideally, given two matching points $r_i \in r$ and $s_j \in s$, s_j (r_i, respectively) should contribute to $LCSS_{\epsilon_t,\epsilon_{sp}}(r,s)$, proportionally to the distance $d_s(r_i,s_j)$. For this reason, we propose a "weighted" LCSS similarity between trajectories, that incorporates the aforementioned distance proportionality. In more detail, for each discovered longest common subsequence the similarity is defined as:

$$Sim(r,s) = \frac{\sum_{k=1}^{min(|r|,|s|)} (1 - \frac{d_s(r_k,s_k)}{\epsilon_{sp}})}{min(|r|,|s|)} \tag{2}$$

where (r_k, s_k) is a pair of matched points.

3.3.3.2 A Closer Look to the Subtrajectory Clustering Problem

Our approach to subtrajectory clustering splits the problem in three steps. The first step is to retrieve for each trajectory $r \in D$, all the moving objects, with their respective portion of movement, that moved close enough in space and time with r, for at least some time duration. Actually, this first step

is a well-defined problem in the literature of mobility data management, known as *subtrajectory join*, and more specifically the case of self-join. In detail, the subtrajectory join will return for each pair of (sub)trajectories, all the common subsequencies that have at least some time duration, which are actually candidates for the longest common subsequence. Formally:

Problem 1 ***(Subtrajectory Join)*** *Given a temporal tolerance ϵ_t, a spatial threshold ϵ_{sp} and a time duration δt, retrieve all pairs of subtrajectories $(r', s') \in D$ such that: (a) for each pair $\Delta t_{r'}, \Delta t_{s'} \geq \delta t$, (b) $\forall r_i \in r'$ there exists at least one $s_j \in s'$ so that $d_s(r_i, s_j) \leq \epsilon_{sp}$ and $d_t(r_i, s_j) \leq \epsilon_t$, and (c) $\forall s_j \in s'$ there exist at least one $r_i \in r'$ so that $d_s(s_j, r_i) \leq \epsilon_{sp}$ and $d_t(s_j, r_i) \leq \epsilon_t$.*

The second step takes as input the result of the first step, which is actually a trajectory r and its neighboring trajectories and aims at segmenting each $r \in D$ into a set of subtrajectories. The way that a trajectory is segmented into subtrajectories is neighbourhood-aware, meaning that a trajectory will be segmented every time its neighbourhood changes significantly. Returning to Example 1, trajectory $A \rightarrow D$ should be segmented to $A \rightarrow O$ and $O \rightarrow D$, since at O the cardinality and the composition of its neighbourhood changes significantly. The problem of trajectory segmentation can now be formulated as follows.

Problem 2 ***(Trajectory Segmentation)*** *Given a trajectory r, identify the set of timestamps CP (cutting points), where the density (or alternatively the composition) of the neighborhood of r changes significantly. Then according to CP, r is partitioned to a set of subtrajectories $\{r'_1, \ldots, r'_M\}$, where $M = |CP| + 1$ is the number of subtrajectories for a given trajectory r, such that $r = \bigcup_{k=1}^{M} r'_k$ and $k \in [1, M]$.*

Given the output of Problem 1, applying a trajectory segmentation algorithm for the trajectories D will result in a new set of subtrajectories D'. The third step takes as input D' and the goal is to create clusters (whose cardinality is unknown) of similar subtrajectories and at the same time identify subtrajectories that are significantly dissimilar from the others (outliers). More specifically, let $C = \{C_1, \ldots, C_K\}$ denote the clustering, where K is the number of clusters, and for every pair of clusters C_i and C_j, with $i, j \in [1, K]$, it holds that $C_i \cap C_j = \emptyset$. Now, let us assume that each cluster $C_i \in C$ is represented by one subtrajectory $R_i \in C_i$, called *Representative*. Furthermore, let R denote the set of all representatives. Actually, the problem of clustering is to discover clusters of objects such that the intra-cluster similarity is maximized and the inter-cluster similarity is minimized. Therefore, if we ensure that the similarity between the representatives is zero, then the problem of subtrajectory clustering can be formulated as an optimization problem as follows.

Problem 3 ***(Subtrajectory Clustering and Outlier Detection)*** *Given a set of subtrajectories D', partition D' into a set of clusters C and a set of outliers O, where $D' = C \cup O$, in such a way so that the Sum of Similarity between Cluster members and cluster Representatives*

*(*SSCR*) is maximized:*

$$SSCR = \sum_{\forall R_i \in R} \sum_{\forall r'_j \in C_i} Sim(R_i, r'_j) \tag{3}$$

However, trying to solve Problem 3 by maximizing Equation (3) is not trivial, since the problem to segment trajectories to subtrajectories, select the set of representatives R and its cardinality $|R|$ that maximizes Equation (3), has combinatorial complexity.

In this Section, we address the challenging problem of subtrajectory clustering in a distributed setting, where the dataset D is distributed across different nodes, and centralized processing is prohibitively expensive.

Problem 4 ***(Distributed Subtrajectory Clustering)*** *Given a distributed set of trajectories, $D = \cup_{i=1}^{P} D_i$, where P is the number of partitions of D, perform the subtrajectory clustering task in a parallel manner.*

Actually, Problem 4 can be broken down to solving Problems 1, 2 and 3 (in that order) in a parallel/distributed way. In the following, we adopt this approach and outline a solution that is based on MapReduce.

3.3.4 Problem Solution

3.3.4.1 Overview An overview of our approach is presented in Algorithm 1.

Algorithm 1 $DSC(D)$

1: **Input:** D
2: **Output:** set C of clusters, set O of outliers
3: **Preprocessing:** *Repartition* D;
4: **for each** partition $D_i \in \cup_{i=1}^{P} D_i$ **do**
5: **perform** *Point-level Join*;
6: **group by** *Trajectory*;
7: **for each** Trajectory $r \in D$ **do**
8: **perform** *Subtrajectory Join*; – *Sect. 3.3.4.2*
9: **perform** *Trajectory Segmentation*; – *Sect. 3.3.4.3*
10: **group by** D_i;
11: **for each** subtrajectory $r' \in D_i$ **do**
12: **calculate** $Sim(r', s')$ $\forall s' \in D_i$; – *Sect. 3.3.4.3*
13: **perform** *Clustering*; – *Sect. 3.3.4.4*
14: **perform** *Refine Results*;
15: **return** C and O;

Initially, we *Repartition* the data into P equi-sized, temporally-sorted temporal partitions (files), which are going to be used as input for the join algorithm in order to perform the subtrajectory join in a distributed way (line 3). Note that this is actually a preprocessing step that only needs to take place once for each dataset D. However, it is essential as it enables load balancing, by addressing the issue of temporal skewness in the input data. Subsequently, for each

partition $D_i \in \cup_{i=1}^{P} D_i$ and for each trajectory we discover parts of other trajectories that moved close enough in space an time (line 5). Successively, we group by trajectory in order to perform the subtrajectory join (line 8). At this phase, since our data is already grouped by trajectory, we also perform trajectory segmentation in order to split each trajectory to subtrajectories (line 9). In turn, we utilize the temporal partitions created during the *Repartition* phase and re-group the data by temporal partition. For each $D_i \in \cup_{i=1}^{P} D_i$ we calculate the similarity between subtrajectories and perform the clustering procedure (line 12). At this point we should mention that if a subtrajectory intersects the borders of multiple partitions, then it is replicated in all of them. This will result in having duplicate and possibly contradicting results. For this reason, as a final step, we treat this case by utilizing the *Refine Results* procedure (line 14). Finally, a set C of clusters and a set O of outliers are produced.

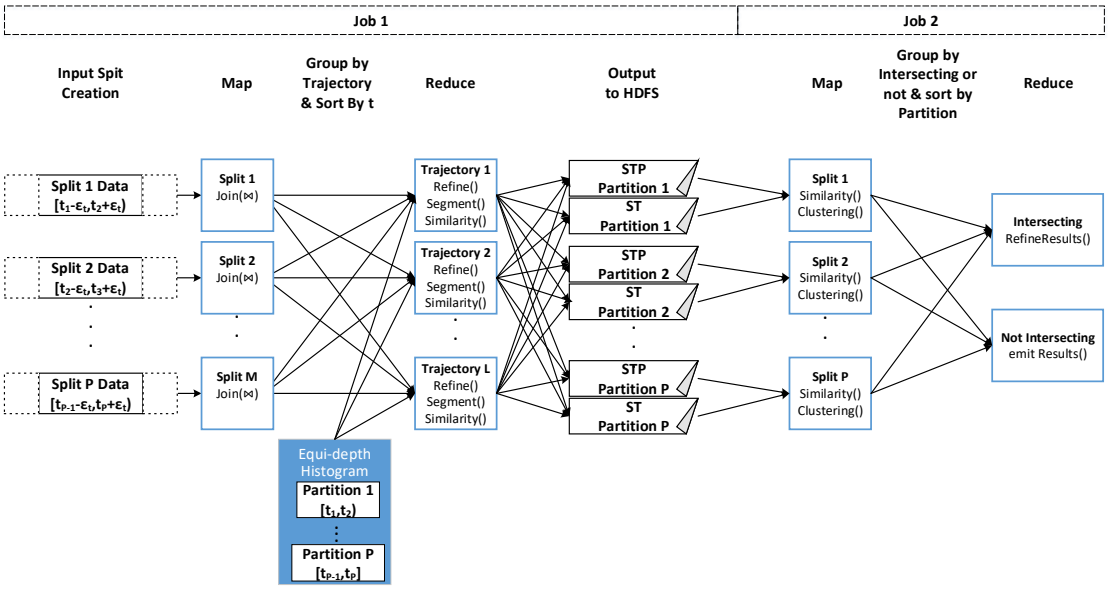

Figure 18: The *DSC* algorithm. (Job 1) *DTJ* and *Trajectory Segmentation* and (Job 2) *Clustering* and *Refine Results*.

3.3.4.2 Distributed Subtrajectory Join As already mentioned, the first step is to perform the subtrajectory join in a distributed way. For this reason, we exploit the work presented in [170], called *DTJ*, which introduces an efficient and highly scalable approach to deal with Problem 1, by means of MapReduce. More specifically, *DTJ* is comprised of a *Repartitioning* phase and a *Query* phase. The *Repartitioning* phase is a preprocessing step that takes place only once and it is independent of the actual parameters of the problem, namely ϵ_{sp}, ϵ_t, and δt. The idea is to construct an equi-depth histogram based on the temporal dimension, where each of the M bins contain the same number of points and the borders of each bin correspond to a temporal interval $[t_i, t_j)$. The histogram is constructed by taking a sample of the input data Then, the

input data is partitioned to processing tasks based on the temporal intervals of the histogram bins. This guarantees temporal locality in each partition, as well as equi-sized partitions, thus balancing the load fairly.

In the *Query* phase, the actual join processing takes place. It consists of two steps, the *Join* (line 5) and the *Refine* (line 8) step, which are implemented as a *Map* and a *Reduce* function respectively. The output of this MapReduce job is for each trajectory $r \in D$ all the moving objects, with their respective portion of movement, that moved close enough in space and time for at least some time duration. In more detail, the output of DTJ is per trajectory (i.e reference trajectory) and the tuples are of the form $< refTrajPoint_i, \{MatchingPoints\} >$, where $refTrajPoint_i$ is the i-th point of the reference trajectory, with $i \in [1, N]$ and $MatchingPoints$ is a list of points of other trajectories that have been identified as join results by the DTJ query. In Figure 18, the DTJ query corresponds to Job 1 until the Refine() procedure.

The complexity of the *Join* algorithm is $O(|D|log_2 Q)$, with Q being the average number of points per spatial index partition and $Q << |D|$. The complexity of the *Refine* algorithm is $O(T \cdot SW \cdot dt \cdot l)$, where T is the average number of points per trajectory, SW is the average number of points contained in a $\delta t + 2\epsilon_t$ window, dt the average number of points contained in a δt window and l is average the size of the $MatchingPoints$ list. For more technical details about the algorithms involved in DTJ, their complexity and an extensive experimental study, we refer to [170].

3.3.4.3 Distributed Trajectory Segmentation The *Trajectory Segmentation* algorithm (TSA) takes as input a single trajectory, along with information about its neighborhood, and partitions it to a set of subtrajectories. Here, we propose two alternative segmentation algorithms. The first algorithm, coined TSA_1, identifies the beginning of a new subtrajectory whenever the density of its neighborhood changes significantly. Such a segmentation algorithm is reminiscent of the flock definition [95], where the identified groups need to be composed of at least m objects. For this purpose, we use the concept of *voting* as a measure of density of the surrounding area of a trajectory. For a given point r_i and any trajectory s, the voting $V(r_i)$ is defined as:

$$V(r_i) = \sum_{\forall s \in D} \frac{d_s(r_i, s_k)}{\epsilon_{sp}} \tag{4}$$

where, s_k is the matching point of s with r_i, as emitted by the subtrajectory join procedure. For a trajectory r that consists of N points $\{r_1, \ldots, r_N\}$, we compute its normalized voting vector $\overline{V}(r)$ as follows:

$$\overline{V}(r)[] = \{\frac{V(r_1)}{\max_{i=1}^{N} V(r_i)}, \ldots, \frac{V(r_N)}{\max_{i=1}^{N} V(r_i)}\} \tag{5}$$

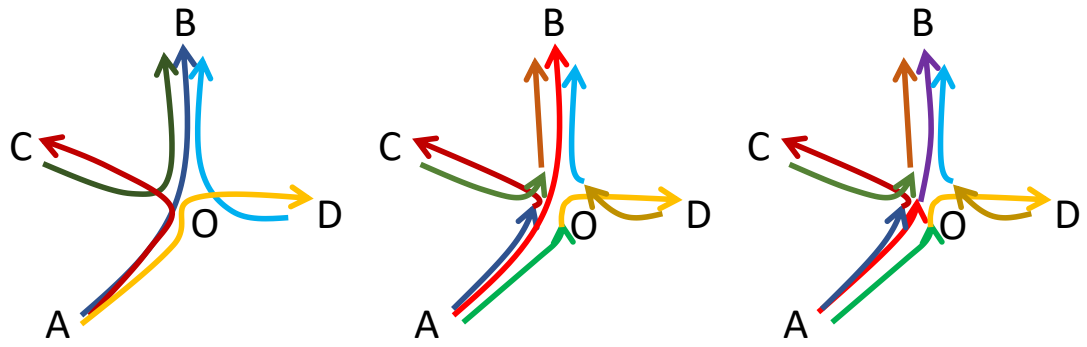

Figure 19: (a) Five trajectories $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $C \rightarrow B$ and $D \rightarrow B$, (b) TSA_1 segmentation, (c) TSA_2 segmentation

Finally, the voting of a trajectory (or subtrajectory) is defined as:

$$V(r) = \frac{1}{N} \sum_{i=1}^{N} V(r_i) \tag{6}$$

The second segmentation algorithm, coined TSA_2, identifies the beginning of a new subtrajectory whenever the composition of its neighborhood changes substantially. This segmentation algorithm is reminiscent of the moving cluster definition [90], where the identified groups need to share a sufficient number of common objects. Such an algorithm does not take as input the $\overline{V}(r)[]$ but instead, for each point $r_i \in r$, it takes as input a list $L(r_i)[]$ of the trajectory ids that have been produced as output by the DTJ procedure.

The following example explains intuitively the difference between the two segmentation algorithms.

Example 2 *Consider the example of Figure 19(a) that illustrates five trajectories: $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $C \rightarrow B$ and $D \rightarrow B$. Figures 19(b) and (c) depict the result of TSA_1 and TSA_2, respectively. In more detail, we can observe that both TSA_1 and TSA_2 segmented trajectory $A \rightarrow D$ to subtrajectories $A \rightarrow O$ and $O \rightarrow D$, due to the fact that after O, both the density and the composition of the neighborhood changes. The same holds for trajectories $A \rightarrow C$, $C \rightarrow B$ and $D \rightarrow B$, which are segmented to subtrajectories $A \rightarrow O$, $O \rightarrow C$, $C \rightarrow O$, $O \rightarrow B$, $D \rightarrow O$ and $O \rightarrow B$. However, when it comes to trajectory $A \rightarrow B$, we can observe that while TSA_2 segments it to subtrajectories $A \rightarrow O$ and $O \rightarrow B$, TSA_1 does not perform any segmentation. This is due to the fact that, after O, even though the density of the neighborhood remains the same (i.e. 3 moving objects), the composition of the neighborhood changes completely.*

Both segmentation algorithms share a common methodology, which employs two consecutive sliding windows W_1 and W_2 of size w (i.e. w samples) to estimate the point $r_i \in CP$ (cutting point) where the "difference" between the two windows is maximized. This methodology has

been successfully applied in the past on signal segmentation [135, 133].

Trajectory segmentation. Since the output of the DTJ algorithm is per trajectory, it is straightforward to give it as input to TSA which operates at the level of a trajectory. Moreover, the segmentation is performed in an embarrassingly parallel way, due to the fact that each trajectory can be processed by a different reduce task independently from others, as depicted in Figure 18. In more detail, for a given trajectory $r \in D$, TSA_1 first calculates the normalized voting vector $\overline{V}(r)[]$ and then performs the segmentation by utilizing it. Apart from $\overline{V}(r)[]$, the input of the TSA algorithm is two additional parameters: w and τ. The output is a vector $CP[]$, which keeps the starting position of each subtrajectory of r.

Algorithm 2 $TSA_1(\overline{V}(r)[], w, \tau)$

1: **Input:** $\overline{V}(r)[], w, \tau$
2: **Output:** $CP[]$
3: $1 \rightarrow CP[]$;
4: **for** n = w+1 ... N-w-1 **do**
5: $m_1 = \frac{1}{w} \sum_{i=n-w}^{n-1} \overline{V}(r)[i]$;
6: $m_2 = \frac{1}{w} \sum_{i=n}^{n+w-1} \overline{V}(r)[i]$;
7: $d[n] = |m_1 - m_2|$;
8: $d_{max} = \max_{i=w+1}^{N-w-1} d[i]$;
9: **if** $d[n] > \tau \wedge d[n] >= d_{max}$ **then**
10: $n \rightarrow CP[]$;

In more detail, as presented in Algorithm 2, two consecutive sliding windows of size w are created over $\overline{V}(r)[]$, named W_1 and W_2 (line 4). These sliding windows move forward in time until $\overline{V}(r)[]$ is traversed. Here, N is the number of points of trajectory $r \in D$. Then, for each window, the average normalized voting is computed (lines 5-6) and their absolute difference is stored in $d[]$, which is an array that stores all the differences between the sliding windows (line 7). Subsequently, we examine whether the current difference $d[n]$ is larger than the maximum difference d_{max} and we update d_{max} accordingly (line 8). Finally, if the difference $d[n]$ is higher than a threshold τ and is locally maximized, then, at that point, we segment the trajectory and we store the starting position of the new subtrajectory to $CP[]$ (lines 9-10).

On the other hand, the input of TSA_2 is a list of lists $L(r)[]$ for each $r \in D$. Similarly, two consecutive sliding windows W_1 and W_2 of size w are created. Then, for each window, the union of lists is computed and stored in l_1 and l_2, respectively. Successively, the Jaccard dissimilarity between l_1 and l_2 is computed and is stored to $d[]$, which is an array that stores all the similarities between the sliding windows. From then on, the algorithm is identical to TSA_1.

The complexity of both TSA_1 and TSA_2 algorithms is $O(l \cdot |T|)$, where l is average the size of the "matching" list and $|T|$ is the average number of points per trajectory.

Similar subtrajectories. The next step is to calculate the similarity between all the pairs of subtrajectories, using Equation 2. This cannot be done completely after the segmentation at the *Reducer* phase of Job 1, illustrated in Figure 18, because at that point each reduce function

has information only about the segmentation of the reference trajectory to subtrajectories. For this reason, at this point we cannot calculate the denominator of Equation 2. However, for each subtrajectory $r' \in r$, where r is the reference trajectory, we can calculate the similarity between the matching points (enumerator of Equation 2).

In more detail the output of each reduce function (Job 1 Figure 18) is a relation, called STP, which holds a set of key-value pairs of the form $< (r'.ID, s.ID), \{(s_f.t, Sim(s_f, r')) \ldots (s_l.t, Sim(s_l, r'))\} >$, where s_f, s_l are the temporal first and last point, respectively, of trajectory s that "matches" with subtrajectory r'. Moreover, in a separate relation, coined ST, we hold some extra information for each subtrajectory. More specifically, the tuples of ST are key-value pairs, where the key is the subtrajectory identifier $< ID >$ and the value is of the form $< t_s, t_e, V, Card >$, where t_s (t_e) is the starting time (ending time, respectively) of the subtrajectory, V is the voting and $Card$ is the number of points which constitute the specific subtrajectory. Due to the fact that these two relations can be pretty large, we need to partition them into smaller files. In order to achieve this, we broadcast the load balanced temporal partitions that were created during the *Repartitioning* phase of DTJ. As illustrated in Figure 18, each reducer loads these partitions and assigns each subtrajectory (tuple of ST and STP) to all the partitions with which it temporally intersects. Subsequently, the tuples are grouped by temporal partition and each group is fed to a Mapper.

At this point, each *Mapper* has now all the information needed to calculate the similarity between all the pairs of subtrajectories (Equation 2), for each temporal partition separately. The similarity between subtrajectories is output in a new relation, called SP. Each tuple of this relation holds information about a subtrajectory r' and its similarity with all the other subtrajectories, whenever this similarity is larger than zero. More specifically, SP contains a set of key-value pairs where the key is the ID of the subtrajectory ($r'.ID$) and the value is a list $AdjLst$ containing elements of the form $(s'.ID, Sim)$, where s' is a subtrajectory for which it holds that $Sim(r', s') > 0$.

3.3.4.4 Distributed Clustering Clustering. After having calculated the similarity between all pairs of subtrajectories for each temporal partition, we can proceed to the actual clustering and outlier detection procedure. The intuition behind the proposed solution to Problem 3 is to select as cluster representatives, highly voted subtrajectories (Equation 6) that have zero similarity with the already selected representatives $R_i \in R$, thus addressing the inter-cluster distance minimization. Then, we assign each subtrajectory r'_k to the R_i (and hence C_i) with which it has the maximum similarity $Sim(r'_k, R_i)$.

The input of the clustering algorithm is SP, ST and parameters k and α and the output is the set of clusters C and the set of outliers O. More specifically, k is a threshold for setting a lower bound on the voting of a representative. This prevents the algorithm from identifying clusters with small support. Parameter α is a similarity threshold used to assign subtrajectories to cluster representatives. It ensures that a subtrajectory assigned to a cluster has sufficient

Algorithm 3 $Clustering(SP, ST, k, \alpha)$

```
1:  Input: SP, ST, k, α
2:  Output: set C of clusters, set O of outliers
3:  sort ST by V in descending order;
4:  for each element st ∈ ST do
5:    if st ∉ R then
6:      if st.V ≥ k then
7:        st → R;
8:        for each element l ∈ st.AdjLst do
9:          if l ∉ C then
10:           if Sim(l, st) ≥ α then
11:             l → C(st);
12:             if l ∈ O then
13:               O = O − l;
14:           else
15:             O = O ∪ l
16:         else
17:           if Sim(l, st) > Sim(l, R(l)) then
18:             C(R(l)) = C(R(l)) − l;
19:             l → C(st);
20:     else
21:       O = O ∪ st;
22: C = C ∪ R
```

similarity with the representative of the cluster. This actually poses a lower bound to the average distance between the representatives and the cluster members and, consequently, guarantees a minimum quality in the identified clusters (intra-cluster distance).

To begin with, we want to traverse the subtrajectories by their voting, in descending order (i.e. highly voted subtrajectories first). In order to achieve this, we need to sort ST by V (line 3). Subsequently, for each subtrajectory $st \in ST$ we examine whether it is already assigned to cluster (line 5). If st is not assigned to any cluster and the voting of st is less than k, then we add st to the outliers set (line 21). Otherwise, we create a new cluster and consider st as the representative (lines 6-7). Successively, we consult relation SP and retrieve the adjacency list of st (line 8). Then, for each element l that belongs to the adjacency list of st, we examine if it is assigned to any cluster. If not, we investigate whether the similarity between l and st is greater or equal than the similarity threshold α. If not, we add l to the outlier set O, otherwise we assign it to the cluster led by st and remove it from the outliers O, in case $l \in O$ (lines 9-13). If l is assigned to a cluster, we examine whether the similarity of l with st is greater than the similarity with the representative of the cluster that l is currently assigned. If this is the case, then we remove l from the current cluster and assign it to the cluster led by st (lines 17-19). Finally, we concatenate C with R (line 22) so as to return, except from the outlier set O, both cluster members and representatives.

Refinement of Results. At this point we successfully accomplished to deal with Problem

3 for each temporal partition. However, this might result in having duplicates due to the fact that each subtrajectory that temporally intersects multiple partitions is replicated to each one of them. The actual problem that lies here is not the duplicate elimination problem itself but the fact that the result for such a subtrajectory might be contradicting in different partitions. In more detail, for each partition, the clustering procedure will decide whether a subtrajectory is a *Representative* (R), a *Cluster Member* (C) or an *Outlier* (O). Hence, for each intersecting subtrajectory q and for each pair of consecutive partitions (i, j) with which q intersects, q can have the following pairs of states: (a) O-O, (b) R-R, (c) C-C, (d) R-C (C-R), (e) R-O (O-R) and (f) C-O (O-C).

In order to implement the above procedure we need to have all the information concerning the intersecting subtrajectories (C and O) for all the Partitions sorted in time. To do this, we group the trajectories according to whether they are intersecting or not. As illustrated in Figure 18, the non-intersecting are emitted, since they are not affected, while the intersecting subtrajectories get sorted by partition. Hence, a *Reducer* will receive all the required information to make the appropriate decisions. In more detail, we sweep through the temporal dimension and for each pair of consecutive partitions we make the appropriate decisions.

More specifically, in case of (a), q is marked as outlier in both partitions, hence, we only need to eliminate duplicates. In case of (b), the two clusters are "merged", since all of the subtrajectories that belong to them are similar "enough" with q, which is the representative of both clusters. In case of (c), let us assume that q belongs to cluster $C_i(R(q))$ in Partition i and $C_{i+1}(R(q))$ in Partition $i+1$. Then, q is assigned to the cluster with which it has the largest similarity with its representative. In case of (d), q remains to be a cluster representative and is removed from the cluster C in which it is a member. Finally, in case of (e) and (f), q is removed from O.

The complexity of the *Clustering* algorithm is $O(|ST| \cdot log|ST| + |ST| \cdot |L|)$, with $|ST|$ being the number of subtrajectories, $|L|$ the average size of the adjacency list $AdjLst$ and $|ST| \cdot log|ST|$ is the sorting cost. Here, we should mention that $|ST| << |D|$. Furthermore, ST and SP are implemented as HashMaps, hence key search has an $O(1)$ time complexity. The complexity of the $RefineResults$ algorithm is $O(M \cdot |P| \cdot |I|)$, where M is the number of temporal partitions, $|P|$ is the average number of intersecting subtrajectories per partition and I is the average size of the intersection. We should mention, here, that the intersection between two consecutive partitions is performed in linear time by utilizing HashSets sets.

3.3.5 Experimental Study

In this section, we present the findings of our experimental evaluation. The experiments were conducted in a 49 node Hadoop 2.7.2 cluster, provided by *okeanos*[1]. The master node consists of 8 CPU cores, 8 GB of RAM and 60 GB of HDD while each slave node is comprised of 4 CPU cores, 4 GB of RAM and 60 GB of HDD. Our configuration enables us to launch up to 192 tasks

[1]IAAS for the Greek Academic Community https://okeanos.grnet.gr/home/

Table 3: Parameters and default values (in bold)

Parameter	**Values**				
	(i)	(ii)	(iii)	(iv)	(v)
ϵ_{sp} (%)	10%	15%	**20%**	25%	30%
ϵ_t (%), δt (%)	0%	25%	**50%**	75%	100%
w	10	15	**20**	25	30
τ	0.2	0.4	**0.6**	0.8	1
α (in σ), k (in σ)	-2	-1	**0**	1	2

simultaneously.

For our experimental study, we employed a real dataset from the urban domain (urban and the maritime). In more detail, the real dataset, named SIS[2], is a 27GB proprietary insurance dataset of moving objects around Rome and Tuscany area, that contains approximately 2.2×10^7 trajectories that correspond to 7.2×10^8 points.

Our experimental methodology is as follows: Initially, in Section 3.3.5.2 we compare our solution with two state of the art subtrajectory clustering methods (TraClus [98] and S^2T-Clustering [144]). Subsequently, in Section 3.3.5.3, we study the scalability of our solution by varying (a) the dataset size, and (b) the number of cluster nodes. Finally, in Section 3.3.5.4, we perform a sensitivity analysis in order to evaluate the effect of setting different values to the parameters, in terms of execution time and quality. Table 3 shows the experimental setting, where we vary the following parameters: ϵ_{sp}, ϵ_t, δt, w, τ, α and k and measure their effect in the performance and the effectiveness of our algorithms. We should mention that the default segmentation algorithm in our experimental study is TSA_1.

3.3.5.1 Parameter Setting Setting the different parameters for different datasets can turn out to be an arbitrary procedure, which, in turn, can jeopardise the quality of the clustering results. For this reason, we provide some simple rules for setting the parameters relatively to the dataset being clustered, that do not compromise the quality of the results. In more detail, ϵ_{sp} can be set as a percentage of the dataset diameter. This, however, can be problematic when dealing with datasets having large spatial variation in their density (e.g. ports in the maritime domain). For this reason, we utilized the partitioning provided by the spatial index (QuadTree) of DTJ and calculated ϵ_{sp} for each point, as a percentage of the diameter of the cell of the QuadTree to which it belongs. Moreover, ϵ_t and δt are calculated relatively to the average duration between two consecutive trajectory samples ($\approx$ 1200 sec for SIS and $\approx$ 950 sec for AIS Brest).

Concerning parameter w, small values on w can affect the robustness of the estimation, thus resulting to over-segmentation, while, large values of w can result to overlooking some cutting points due to the large window size. It has been observed that for $w \approx 20$ the robustness of

[2]This private dataset was kindly provided by Gruppo Sistematica SpA

the estimation is not affected and the size of the window is small enough so as not to overlook any cutting points. Concerning parameter τ, our experiments show that the best result in terms of quality is achieved for $\tau \approx 0.4$ Finally, the values of α and k can be set "around" the mean value of the similarity and the voting of the temporal partition, respectively, in terms of standard deviation. For more details about the effect, in terms of quality, of setting different values to the parameters of our solution, please refer to Section 3.3.5.4

3.3.5.2 Comparison with related work We compare *DSC* with two state of the art sub-trajectory clustering algorithms, *S^2T-Clustering* and *TraClus*. The metric that we employ in order to evaluate the quality of the outcome of the clustering procedure is the well-known *RMSE* metric, which is actually a measure of intra-cluster distance between the representatives and the cluster members. Hence the larger the *RMSE*, the higher the intra-cluster distance and consequently the lower the quality of the clustering. In order to perform this experiment, we utilized the 20% of each dataset which was further partitioned in 4 portions (25%, 50%, 75%, 100%). This choice was necessary because the centralized implementations of *S^2T-Clustering* and *TraClus* could not scale with the full size of the datasets.

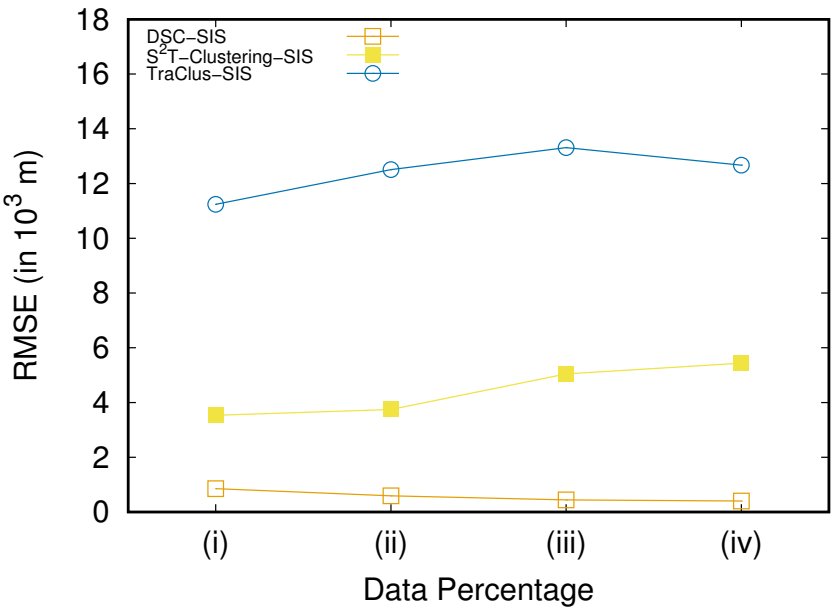

Figure 20: Comparison of the RMSE metric between *DSC*, *S^2T-Clustering* and *TraClus*

As illustrated in Figure 20, *DSC* outperforms, in terms of *RMSE*, both *TraClus* and *S^2T-Clustering*. In more detail, *TraClus* presents the largest *RMSE* which is somehow anticipated, since the specific algorithm utilizes a density-based approach to cluster subtrajectories, which in turn, through cluster expansion, can lead to spatially extended clusters. On the other hand, *S^2T-Clustering* presents smaller *RMSE* than *TraClus*, due to the fact that it adopts a distance-

based approach and discovers more compact clusters. However, *DSC* results in smaller *RMSE* than S^2 *T-Clustering*, mostly due to the fact that in the latter, two trajectories might end-up in the same cluster even if they have small "matching portions". However, in *DSC* this "matching portions" should have a minimum (δt) duration.

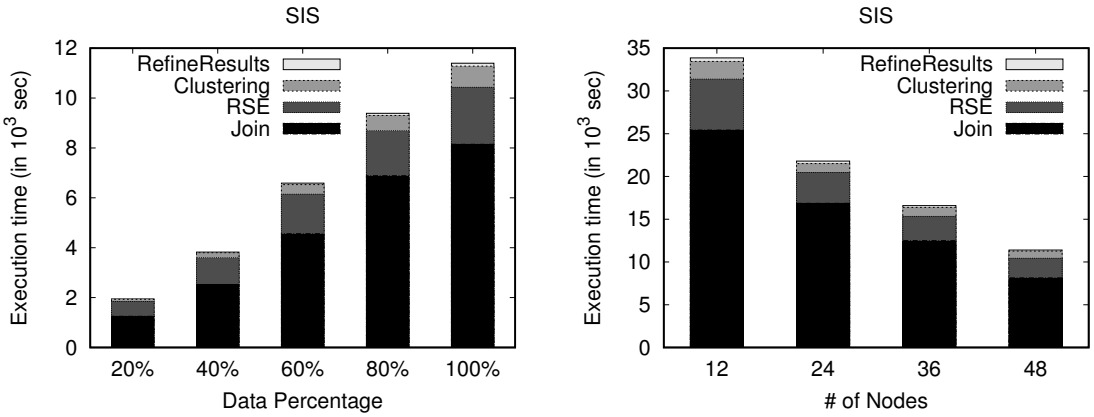

Figure 21: Scalability by varying (a) the size of the dataset and (b) the number of nodes.

3.3.5.3 Performance and Scalability Initially, we vary the size of our datasets and measure the execution time of our algorithms. We show the impact of the individual steps: *Join*, *RSE*, *Clustering* and *RefineResult* using stacked bars. To study the effect of dataset size, we created 4 portions (20%, 40%, 60%, 80%) of the original datasets. *RSE* stands for the *Refine* and *Segmentation* procedure (Figure 18, Job 1, Reduce phase). As illustrated in Figures 21(a) and (b), as the size of the dataset increases, *DSC* appears to scale linearly. Subsequently, we keep the size of the datasets fixed (at 100%) and vary the number of nodes. As the number of nodes increases and the dataset size remains the same, it is expected that the execution time will decrease. Indeed, as depicted in Figures 21(c) and (d), as the number of nodes increases, *DSC* presents linear speedup. This linear behaviour, is somehow anticipated due to the fact that the *DSC* approach is dominated by *DTJ*, in terms of execution time, which presents linear speedup, as shown in [170].

Investigating further the performance, we can observe that the execution time of the whole procedure is dominated by the Join step (Figure 18, Job 1, Map phase), followed by *RSE*. Finally, the *Clustering* and the *RefineResults* step (Figure 18, Job 2) present very good performance, since the computationally intensive part of the similarity matrix calculation has already been done as part of the previous steps.

3.3.5.4 Sensitivity Analysis In this section, we vary each parameter presented in Table 3, while keeping the rest of them in their default value (bold), and we measure their effect in the

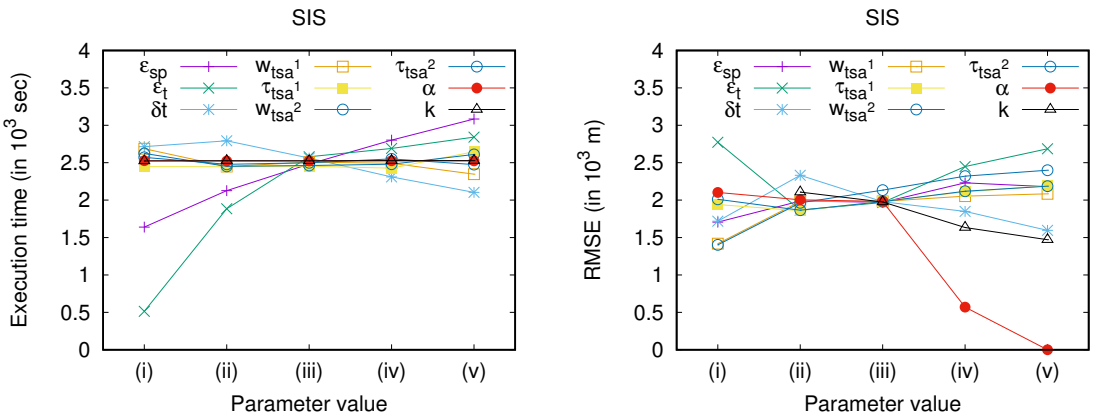

Figure 22: Sensitivity in terms of (a) execution time (b) in terms of *RMSE*.

execution time and the quality of the clustering results, in terms of *RMSE*. Figures 22(a) and (b) show that the parameters that appear to have a significant impact on execution time are ϵ_t and ϵ_{sp}. This is justified from the fact that these parameters actually affect significantly the complexity of the Join step (Figure 18, Job 1, Map phase), which is the dominant cost of *DSC*. Another parameter that seems to have a perceivable effect on the execution time, is δt, which in fact "filters" the results of *DTJ*, thus fewer data reach the next steps.

Regarding the quality of the clustering results, as illustrated in Figure 22(c) and (d), all the parameters seem to have an effect over it. In more detail, the larger the values of ϵ_t and ϵ_{sp}, the larger the *RMSE*. This behaviour is expected since we allow objects that are further away from a representative to participate to the same cluster. In contrast, as δt increases, the *RMSE* decreases, which is also anticipated since it sets a lower bound to the longest common subsequence. Furthermore, all the parameters that control the segmentation have the same effect on the *RMSE*, i.e. the smaller (in length) the subtrajectories, the smaller the *RMSE*. This shows that breaking trajectories to subtrajectories has a positive effect on the quality of the clustering and justifies the motivation of our work. Moreover, as α increases the *RMSE* decreases, since for small values of α, less similar objects are allowed to participate in a cluster. Finally, the larger the k the smaller the *RMSE*, since it disallows the identification of clusters with small support.

3.4 Future Location Prediction (FLP) - Trajectory Prediction (TP)

In summary:

- *Generic question addressed*: Predicting vehicles' future positions.
- *Track&Know specific question*: Future Location Prediction (FLP) for online trajectory analytics.
- *Novelty / Advantage over existing methods*: The proposed methods provide increased prediction accuracy. Also, FLP NN-based is able to predict future positions of vehicles with unknown historical trajectories (that may not be included in the training phase), since it takes into account the patterns made available by the entire population used in the training phase.
- *Experiments conducted*: Experimental evaluation on a Track&Know Pilot dataset with quantitative and qualitative evaluations and comparison to recent state-of-the-art methods. Also, experiments for the run-time efficiency were conducted on the Apache Kafka platform.
- *Type of analytics*: Predictive analytics.
- *Automation / TRL*: TRL level 3 (integrated on Track&Know platform, tested on real data for simulation-based applications, so we are between TRL 3 and 4)
- *Extension to other domains*: Predict the possible evolution of vehicles' movement in order to improve road safety, improve obstacle avoidance and path planning for intelligent vehicles, etc.

Predicting vehicles future locations is valuable to construe events and activities taking place in the urban environment, which is of paramount importance for improving safety. More specifically, it is of high interest to solve the Future Location Prediction (FLP) problem, which, according to Georgiou et.al. [53], aims to predict the next instance position of a vehicle based on its present and previous motion history.

In this work two distinct methods are introduced for solving the FLP problem; the first employs the power of Neural Networks (NNs), while the second is a route-based algorithm.

3.4.1 Part I: NN-based for short-term

Several methods have been proposed, for forecasting temporal and sequential data, while numerous advances have been achieved by using Neural Networks (NNs) for time series forecasting [14]. More specifically, in order to deal with sequential data, a Recurrent NN (RNN) [158] architecture was proposed. RNNs form a dynamic system, where the current state is determined

not only by the input messages, but also, from the previous ones. Thus, RNN-based models keep in memory input information about the delay for an indefinite period of time. On the other hand, traditional NNs, such as the popular Multi-Layer Perceptrons (MLPs), are based on the assumption that their input data are independent of each other [122]. Hence, traditional NNs are not tailored to work with temporal and sequential data [157]. Although, RNNs handle sequential data inherently, they suffer from the well-known vanishing and the exploding gradient issues [74], which hinder the processing of long sequences. To remedy this drawback, a lot of research has been conducted suggesting a number of improvements, with the most successful being the LSTM [74].

Following this trend and as, a rich set of possible future trajectories arises from vehicles' current motion paths, we propose a 'vanilla' LSTM-based NN for the FLP task trained with historical GPS road data derived from a variety of vehicles. This results in a technique capable of forecasting a vehicle's motion path with unknown historical trajectories in the LSTM training phase, i.e., not included in the historic data used in training, as long as they manifest more or less similar movement patterns.

3.4.1.1 Related Work In this paragraph, we review recently introduced vehicle location prediction methods based on NN-based methods, especially RNN models, while extensive surveys on vehicle motion prediction models have been presented by Lefèvre et. al. [99] and Zhan et. al.[198].

Percher et.al. [140] employed various methods, including NNs, to predict taxi-drivers' trajectories, by dividing the road network into a two dimensional grid. Also, in [31] an RNN-based method for urban vehicle trajectory prediction is proposed, where the urban traffic network is partitioned into a grid area composed of cells. However, grid-based segmentation does not take into account the logical cohesion between city areas [122].

Park et.al. [137] provided results for prediction time steps of $\Delta = 0.4, 0.8, 1.2, 1.6$ and 2.0 sec, while in [111] the prediction time is equal to 3 sec. Also, in [91], [77], [7] and [193], the employed maximum prediction horizon is 2sec, 5sec 10 sec and 90sec, respectively. Obviously, these works correspond to short-term prediction.

Wang et.al. [187] proposed an LSTM model for trajectory prediction, which can first make a single-step prediction after one-hour of observation. Furthermore, in [44], the DLNLP was proposed, which predicts vehicle's next location given its trajectories and related contextual information. These are interesting works, however, both of them cannot predict future positions of a vehicle when a decent number of its past positions is not recorded.

3.4.1.2 Problem Formulation Given a dataset composed of trajectories derived from various vehicles and the features of timestamp $t_j^s(k)$, longitude $lon_j^s(k)$ and latitude $lat_j^s(k)$, for each vehicle s, for each trajectory j, for each record-point k, we seek to learn a model that predicts the 2D trajectory composed of longitude $lon_j^s(k+1)$ and latitude $lat_j^s(k+1)$ coordinates at timestamp $t_j^s(k+1)$, which is equal to $t_j^s(k) + \Delta t$.

3.4.1.3 Methodology In human mobility, the most straightforward way to predict the future location is to feed the NN model with massive timestamped locations. Following this trend the proposed vanilla LSTM is fed with historical GPS road data derived from a variety of vehicles. More specifically, the proposed methodology follows 3 phases: Data preparation, NN model application, Data transformation.

In the first phase, the available GPS data are fed to a map-matching procedure, which aligns trajectories to an underlying road network. More specifically, since vehicles are supposed to follow roads, map knowledge employed to estimate vehicles' position by matching vehicles' locations with the corresponding road coordinates. For the purposes of this method, a map-matching technique is applied on the NN input signal in order to decrease GPS positioning error and fluctuations from the actual path [150]. Also, in order to allow Euclidean geometry computations, for each record-point, the geodetic coordinates longitude and latitude are converted into Cartesian coordinates x and y, respectively. Furthermore, to enhance the performance of the LSTM network, the time information together with the coordinates are incorporated to the network by employing the first-order differential processing between two consecutive points described in [112]. To this end, the proposed model, for each vehicle s and each trajectory j, predicts the intervals of UTM longitude and UTM latitude between $k+1$ and k entries, by using an input vector composed of the following elements in the trajectory j of vehicle s:

- The time difference between $k+1$ and k entries, which also indicates the desired time interval that the prediction must take place
- The time difference between k and $k-1$ entries
- The intervals of UTM longitude and UTM latitude, between k and $k-1$ entries

Subsequently, the NN model is employed to predict the coordinates intervals. Finally, the predicted coordinates intervals are being transformed to the actual predicted locations and then a map-matching procedure is employed, which aligns the predicted locations to the underlying road network, due to the fact that the proposed LSTM model assumes nothing about the underlying road network.

A schematic overview of the proposed LSTM-based network architecture is presented in Fig.23.

As far as the employed LSTM model is concerned, for the shake of consistency, we briefly state its update rules below, while for more details, the interested reader is referred to the original publications [74][55]. In the literature there is a number of variants of LSTM architectures. The most commonly used LSTM block, namely vanilla LSTM, includes three gates: forget $\mathbf{f}$, input $\mathbf{i}$, output $\mathbf{o}$. More specifically, an LSTM cell, for each time-step t, is fed with the input vector $\mathbf{u}(t)$ and the updating process can be described by the following equations as also shown in Fig. 24:

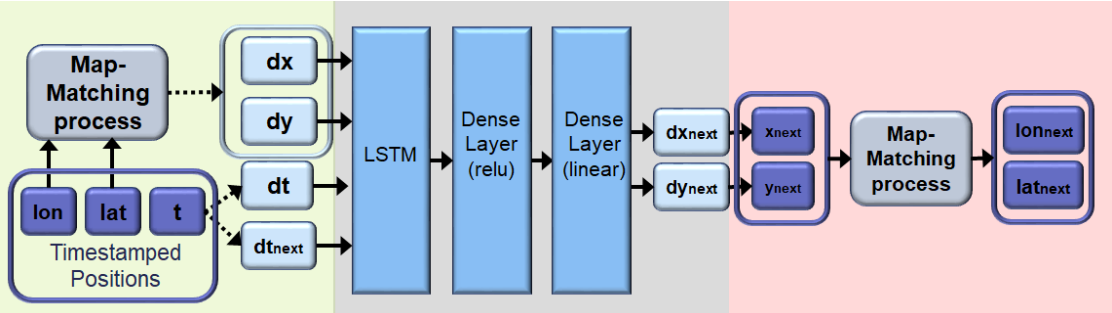

Figure 23: Visualization of the proposed FLP approach, composed of 3 phases: Data preparation (green square), NN model (gray square), Data transformation (red square). The NN is composed of one LSTM cell and two fully connected layers.

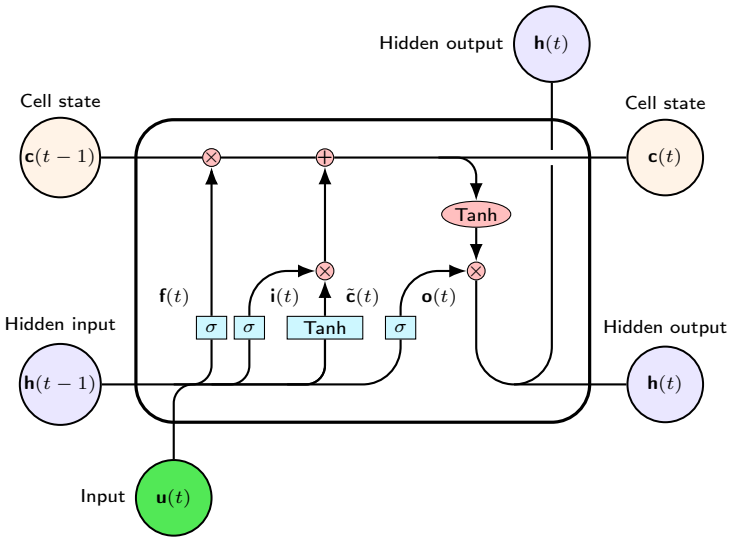

Figure 24: A graphical representation of a vanilla LSTM memory cell

$$
\begin{aligned}
\mathbf{i}(t) &= \sigma(\mathbf{W}_i \cdot \mathbf{u}(t) + \mathbf{R}_i \cdot \mathbf{h}(t-1) + \mathbf{b}_i) \\
\mathbf{f}(t) &= \sigma(\mathbf{W}_f \cdot \mathbf{u}(t) + \mathbf{R}_f \cdot \mathbf{h}(t-1) + \mathbf{b}_f) \\
\mathbf{o}(t) &= \sigma(\mathbf{W}_o \cdot \mathbf{u}(t) + \mathbf{R}_o \cdot \mathbf{h}(t-1) + \mathbf{b}_o) \\
\tilde{\mathbf{c}}(t) &= \tanh(\mathbf{W}_g \cdot \mathbf{u}(t) + \mathbf{R}_g \cdot \mathbf{h}(t-1) + \mathbf{b}_g) \\
\mathbf{c}(t) &= \mathbf{f}(t) \odot \mathbf{c}(t-1) + \mathbf{i}(t) \odot \tilde{\mathbf{c}}(t) \\
\mathbf{h}(t) &= \mathbf{o}(t) \odot \tanh(\mathbf{c}(t))
\end{aligned}
\tag{7}
$$

where:

- $\mathbf{c}$ is the cell state vector,

- $\tilde{\mathbf{c}}$ is the candidate value for the states of the memory cells
- $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_o$, $\mathbf{W}_c$ are the input-to-hidden weight matrices,
- $\mathbf{R}_f$, $\mathbf{R}_i$, $\mathbf{R}_o$, $\mathbf{R}_c$ are the state-to-state recurrent weight matrices,
- $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_o$, $\mathbf{b}_c$ are the bias terms,
- $\odot$ is the Hadamard product (element-wise product) of the vectors,
- $\sigma()$, tanh() are the sigmoid and Hyperbolic tangent functions, respectively.

The proposed approach by taking advantage of the Apache Kafka distributed streaming platform corresponds to an online distributed procedure. More specifically, streaming information from different vehicles is fed to the network at the same time, which a) makes predictions in parallel and b) sends the new information in a distributed streaming way to other tools in the same platform. However, due to invalid points (e.g. GPS noise), the algorithm performs, also, a "validity check", where non-realistic values are being identified and removed. The overall online-architecture based on Apache Kafka is can be seen in Fig.25.

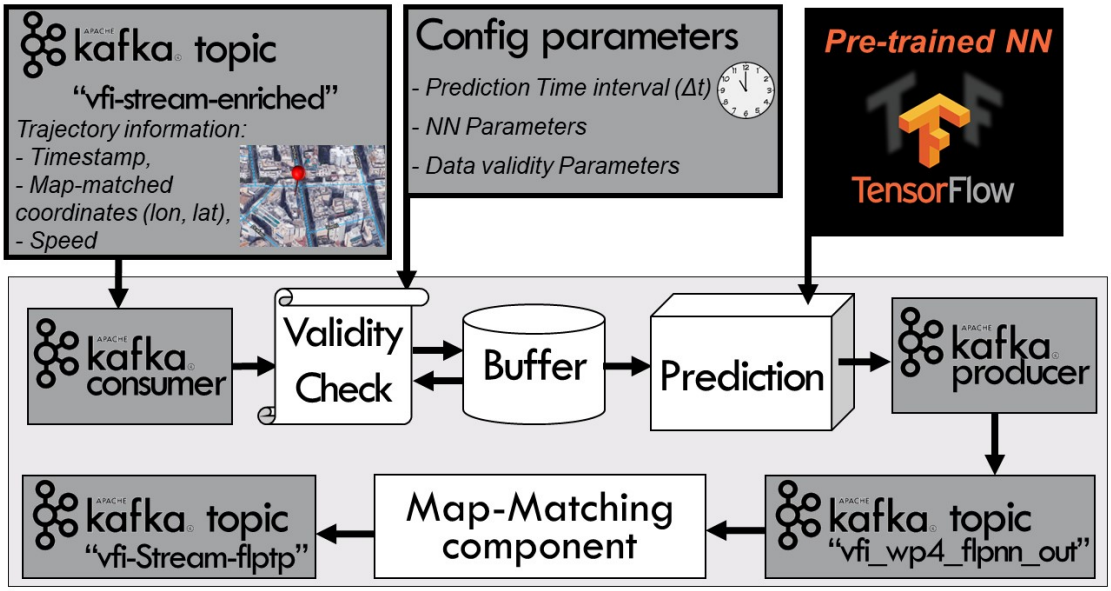

Figure 25: Overview of the FLP online procedure with pre-trained model

3.4.1.4 Experiments The proposed methodology was experimentally evaluated over a sample of the VFI trajectory data. In detail, a total of 977,646 records of location related information derived from GPS data of 2,638 distinct vehicles, during one day (2018/11/02) at Athens

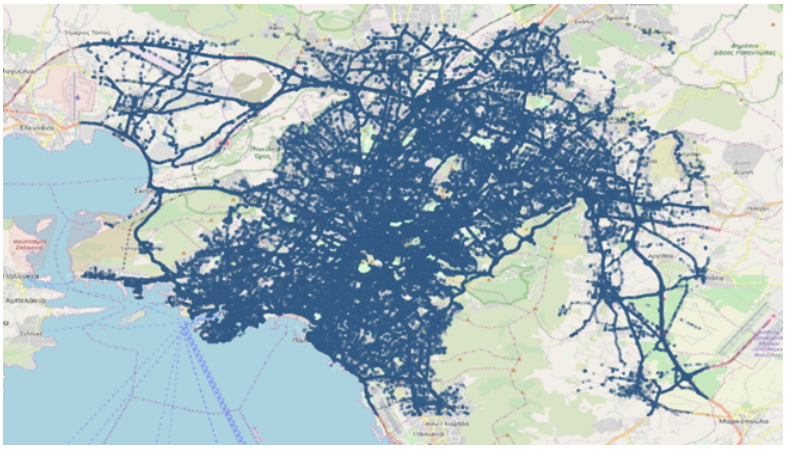

Figure 26: Overview of the employed VFI subset.

region (Attica) with spatial bounding box: Lon: [23.559140, 23.912759] E and Lat: [37.885963, 38.105679] N, as illustrated in Fig.26.

We adopted a method evaluation procedure, similar to [5], i.e. the available trajectories were allocated randomly into three sets: training, validation and testing, employing 50% – 20% – 30% ratio, respectively. The NN parameters were determined using the training set and then model selection was performed using the validation set. Finally, the selected NN model's performance was tested on the testing set, which is independent of training and model selection and thus can assess generalization capabilities. Fig.27 depicts the loss function during NNs training in the training and validation sets.

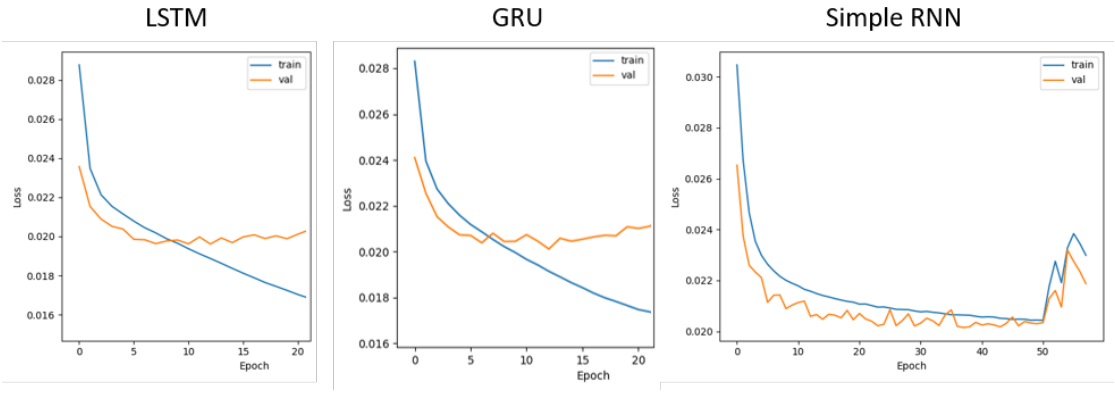

Figure 27: Loss function during training procedure in the training and validation sets for the 3 employed NN models.

The results were evaluated using the Mean Absolute Error (MAE) of the Haversine distance between the original points and the predicted ones on the testing set. Note that, the first 20 points of each trajectory are employed for initializing the NN states; hence, they are not

method	1 min	2 min	3 min	4 min	5 min
LSTM	75	232	262	257	251
GRU	87	245	273	276	273
Vanilla RNN	94	253	305	290	282
MLP+LSTM[190]*	189	386	550	699	798

* Model performance with span equal to 5sec, on the 12, 24, 36, 48 and 60 points for intervals 1, 2, 3, 4, 5 min., respectively.

Table 4: FLP ERROR (Mean Haversine Distance) in meters - Evaluation on VFI subset for certain prediction intervals in minutes.

being considered in the evaluation procedure. Also, for comparison purposes we employed two alternative RNN-based architectures, namely Standard RNN [158] and Gated recurrent units (GRUs) [29]. The alternative NN architectures are similar to the LSTM model architecture, i.e. the Standard RNN model is composed of four input neurons, a Vanilla RNN hidden layer followed by a fully-connected one and two output neurons, while the GRU model comprises the same layers except that the Vanilla RNN is being replaced by a GRU layer.

Moreover, in order to demonstrate the effectiveness of the proposed LSTM model against existing recent state-of-the-art methods, we implemented the proposed "MLP+LSTM" model of [190], while we applied the exact preprocessing procedure proposed in [190]. Particularly, Wang et.al.[190] proposed an encoder-decoder NN, where a linear interpolation method is employed in order to create points of a constant sampling rate of 5 sec. Each trajectory is composed of 1000 timestamps, i.e., 5000 sec., which is being splitted into training and testing subsets. For each interpolated trajectory the initial 800 locations are being used for training purposes and the remaining 200 locations for testing purposes. Then, given the first 30 historical data points the NN model predicts locations in the future n time steps. In order to predict for certain intervals: 1, 2, 3, 4, 5 min., we measure the performance of the 12th, 24th, 36th, 48th and 60th point, respectively.

Fig.28 depicts the results of the LSTM model in the form of boxplots, according to the Haversine distance MAE for certain prediction intervals in minutes. Also, Table 4 depicts results for all employed methodologies, for certain prediction intervals in minutes. Obviously, the LSTM model outperforms the other two NNs, while it predicts satisfactorily the vehicles' next position. Furthermore, in contrast to the existing most recent state-of-the-art method, i.e. the "MLP+LSTM" model proposed in [190], our LSTM model performs 60%, 40%, 52%, 63%, 69% better, for the prediction intervals of 1, 2, 3, 4, 5 min., respectively.

As far as the prediction phase of the LSTM network is concerned, experiments were conducted on a server with 32GB RAM and 8 CPU cores, on the Apache Kafka platform by using 6 consumers, which consume messages from a topic with 6 partitions and the vehicle id as the key. Results can be seen in the following tables. Particularly, Table 5 depicts statistics (minimum,

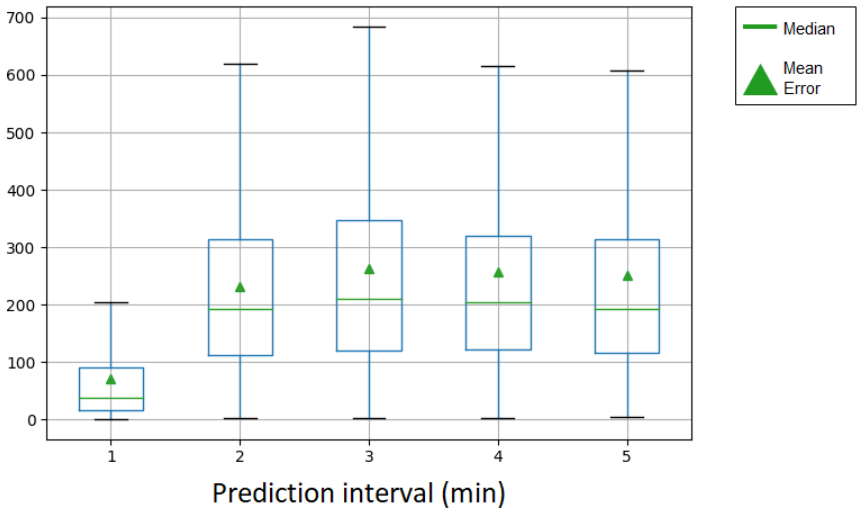

Figure 28: Boxplots for the Haversine distance MAE for certain prediction intervals in minutes for the LSTM model.

maximum, median, mean and standard deviation values from 1000 predictions of each consumer) for the run-time in seconds for one prediction per consumer. Also, Table 6 depicts statistics (minimum, maximum, median, mean and standard deviation values from 6 consumers) for 100, 1000 and 10000 consumed messages, for time in seconds, number of vehicles and number of predictions. Finally, Table 7 depicts statistics (minimum, maximum, median, mean and standard deviation values from 6 consumers) for 1, 10 and 60 seconds, for number of consumed messages, number of vehicles and number of predictions.

Consumer	min	max	median	mean	std
1	0.0259	0.1167	0.0264	0.0354	0.029
2	0.0262	0.1108	0.0269	0.033	0.022
3	0.0261	0.1063	0.0267	0.0306	0.017
4	0.0264	0.1144	0.027	0.0343	0.023
5	0.0263	0.1149	0.0277	0.0341	0.023
6	0.0267	0.1085	0.0282	0.0331	0.02

Table 5: Statistics for the run-time in seconds for one prediction per consumer.

Consumed messages		min	max	median	mean	std
100	Time(sec)	0.2469	0.3373	0.2859	0.2887	0.0373
	Vehicles	80	88	85	85	3
	Predictions	0	3	1	1	1
1000	Time(sec)	2.9059	3.1403	3.1113	3.0655	0.0966
	Vehicles	671	736	719	712	25
	Predictions	12	17	15	15	2
10000	Time(sec)	28.5146	30.1490	29.4780	29.4287	0.5899
	Vehicles	690	756	740	732	25
	Predictions	96	152	135	131	21

Table 6: Statistics for 100, 1000 and 10000 consumed messages, for time in seconds, number of vehicles and number of predictions.

Time(sec)		min	max	median	mean	std
1	Consumed messages	340	269	311	351	351
	Vehicles	77	88	83	82	4
	Predictions	1	10	4	4	3
10	Consumed messages	3121	3216	3260	3384	3215
	Vehicles	508	558	536	534	19
	Predictions	37	59	48	48	8
60	Consumed messages	20932	21349	20975	21053	21105
	Vehicles	684	752	734	727	26
	Predictions	146	211	195	190	23

Table 7: Statistics for 1, 10 and 60 seconds, for number of consumed messages, number of vehicles and number of predictions.

3.4.2 Part II: Pattern-based Future Location Prediction

3.4.2.1 Introduction The "explosion" of trajectory data production due to the proliferation of GPS-enabled devices, poses new challenges in terms of storing, querying, analyzing and extracting knowledge from big mobility data. One of these challenges is to exploit these data by means of identifying historical mobility patterns, which, in turn, can gauge the procedure of discovering what the moving entities might do in the future. As a consequence, predictive analytics over mobility data have become increasingly important and are ubiquitous in many application

scenarios, such as such as collision detection, traffic estimation and service recommendation, in different domains, such as maritime, urban and aviation.

Inspired by this, we propose, a system able to predict simultaneously, in real-time, the exact future location of an extremely large set of moving objects, given a look-ahead time, by employing historical mobility patterns. To do so, we follow a *hybrid* approach, where we a predictor is built for each moving object by considering, when availiable its *individual* past movement and when not, *collective* historical patterns. In this way, we increase the predictive ability of our system, as compared to only using the *individual* history of each moving object and the accuracy of our predictions, as compared to only using *collective* historical patterns.

Solving this problem is quite challenging, since one has to take into account not only the inherent complexity of the *FLP* problem but also the challenges posed by the Big Data era, in terms of volume and velocity of the incoming data. In more detail, the problem can be divided in an *Offline* and an *Online* part. The *Offline* part is responsible for identifying patterns of movement, while the *Online* part is responsible for predicting the future location of a moving object, given a look-ahead time and the set of patterns that were identified during the *Offline* module.

Towards this direction, [147] utilize the work done by [168] on distributed subtrajectory clustering in order to be able to extract individual subtrajectory patterns from big mobility data. These patterns are subsequently utilized in order to predict the future location of the moving objects in parallel. Despite the fact that this solution takes advantage of subtrajectory patterns and is Big Data compliant, it suffers from the fact that it takes account only *individual* patterns, thus decreasing the system's predictive ability. Furthermore, due to the fact that subtrajectory patterns are patterns that are valid for smaller portions of the trajectories lifespan, this might lead to stumbling into "dead ends" (i.e. reaching the end of a pattern and not having the ability to predict further ahead in time), which, in turn, would lead to decreased look-ahead prediction ability. In turn, this might lead to either decreased predictive ability or decreased accuracy, depending on whether the system provides a prediction ([147] do not provide a prediction if the look-ahead time exceeds the lifespan of the pattern) or not ([181] return the last point of the pattern as the predicted point if the look-ahead time exceeds the lifespan of the pattern).

3.4.2.2 Related Work Several efforts try to deal with this problem by applying spatial discretization and generalization and then try to find frequent locations or sequences of locations ([117, 100, 57, 81, 197, 108]). However, such approaches provide generalized and thus inaccurate predictions. Moreover, a large number of approaches do not take into account the temporal information during the mobility behavior extraction and/or during the prediction!([9, 57, 84, 108, 117, 127, 151]). Another line of research, that takes into account both time and the exact location of the moving objects, includes efforts that try to deal with this problem by grouping entire trajectories, identifying patterns of movement and then using them to predict the future location ([181, 100, 54, 81]). However, identifying patterns that are valid for the entire

lifespan of the moving objects can overlook significant patterns that might exist only for some portions of their lifespan. The following motivating example shows the merits of subtrajectory pattern extraction.

Concerning mobility pattern discovery, the aim is to identify several types of collective behavior patterns among moving objects like the so-called flock pattern [95, 185] and the notion of moving clusters [90]. A number of research efforts that emerged from the above ideas are the approaches of convoys [83, 130], platoons [103], swarms [104], gathering pattern [201] and traveling companion [173]. Trasarti et al. [180] introduced "individual mobility patterns" in order to extract the most representative trips of a specific moving object, so that they can predict object's future locations. However, all of the aforementioned approaches are centralized and cannot scale to massive datasets (e.g., at least 10^6). Towards this, the problem of convoy discovery in a distributed environment by employing the MapReduce programming model was studied both in [129]. An approach that defines a new generalized mobility pattern which models various co-movement patterns in a unified way and is deployed on a modern distributed platform (i.e., Apache Spark) to tackle the scalability issue is presented in [43].

Another line of research, tries to discover groups of either entire or portions of trajectories considering their routes. A typical strategy is to transform trajectories to a multi-dimensional space and then apply well-known clustering algorithms such as OPTICS [10] and DBSCAN [40]. Another approach is to define an appropriate similarity function and embed it to an extensible clustering algorithm [123]. Nevertheless, trajectory clustering is an "expensive" operation and centralized solutions cannot scale to massive datasets. Furthermore, [162] proposes a MapReduce approach that aims to identify frequent movement patterns from the trajectories of moving objects. In [78] the authors tackle the problem of parallel trajectory clustering by utilizing the MapReduce programming model and Hadoop. They adopt an iterative approach similar to k-Means in order to identify a user-defined number of clusters, which leads to a large number of MapReduce jobs.

However, discovering clusters of complete trajectories can overlook significant patterns that might exist only for portions of their lifespan. To deal with this, the authors of [98] propose TraClus, a partition-and-group framework for clustering 2-D moving objects which segments the trajectories based on their geometric features, and then clusters them by ignoring the temporal dimension. A more recent approach to the problem of subtrajectory clustering, is S^2T-Clustering [144], where the authors take into account the temporal dimension, and the segmentation of a trajectory takes place whenever the density of its spatiotemporal 'neighborhood' changes significantly. The segmentation phase is followed by a sampling phase, where the most representative subtrajectories are selected and finally the clusters are built "around" these representatives. A similar approach is adopted in [2], where the authors aim at identifying common portions between trajectories,with respect to some constraints and/or objectives, by taking into account the "neighborhood" of each trajectory. These common subtrajectories are then clustered and each cluster is represented by a pathlet, which is a point sequence that is not necessarily a

subsequence of an actual trajectory. A different approach is presented in QuT-Clustering [143] and [171], where the goal is, given a a temporal period of interest W, to efficiently retrieve already clustered subtrajectories, that temporally intersect W. To achieve this, a hierarchical structure, called ReTraTree (Representative Trajectory Tree) that effectively indexes a dataset for subtrajectory clustering purposes, is built and utilized.

3.4.2.3 Problem Definition In this section we are going to provide some preliminary definitions. In more detail, given a set D of moving object trajectories, a trajectory $r \in D$ is a sequence of timestamped locations $\{r_1, \ldots, r_N\}$. Each $r_i = (loc_i, t_i)$ represents the i-th sampled point, $i \in 1, \ldots, N$ of trajectory r, where N denotes the length of r (i.e. the number of points it consists of). Moreover, loc_i denotes the spatial location (2D or 3D) and t_i the time coordinate of point r_i, respectively. A subtrajectory $r_{i,j}$ is a sub-sequence $\{r_i, \ldots, r_j\}$ of r which represents the movement of the object between t_i and t_j where $i < j$ and $i, j \in 1, \ldots, N$. Let $d_s(r_i, s_j)$ denote the spatial distance between two points $r_i \in r$, $s_j \in s$. In our case we adopted the Euclidean distance, however, other metric distance functions might be applied. Also, let $d_t(r_i, s_j)$ denote the temporal distance, defined as $|r_i.t - s_j.t|$. Furthermore, let Δt_r symbolize the duration of trajectory r (similarly for subtrajectories).

Our approach to prediction-oriented subtrajectory pattern network extraction can be split into two sub-problems. The first step is the *subtrajectory pattern extraction*, where the goal is to identify popular patterns of movement and the second step is the *subtrajectory pattern network reconstruction* problem, where the goal is to construct a spatiotemporal directed graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, that represents a network of movement through which a moving object can be routed. For the problem of *subtrajectory pattern extraction* we are going to utilize the work done in Section 3.3. More specifically, the identified cluster representatives are going to play the role of the patterns.

Due to the hybrid nature of out system, there is the possibility that an individual pattern might be identical with a collective pattern. Furthermore, a pattern might be the continuation of another pattern, irrespective of whether they are individual or collective. Hence, an algorithm that takes as input a set of patterns and tries to construct a directed graph, would have to perform a series of "merge" and "append" operations.

Problem 5 ***(Pattern Network Reconstruction)*** *Given a set of patterns R, a spatial threshold ϵ_{sp}, a temporal tolerance ϵ_t and a similarity threshold α, construct a spatiotemporal directed graph $G = (V, E)$ after performing all the appropriate "merge" and append operations.*

In this report, we address the challenging problem of prediction-oriented subtrajectory pattern network extraction in a distributed setting, where the dataset D is distributed across different nodes, and centralized processing is prohibitively expensive. Finally, the Future Location Prediction problem can be defined as follows:

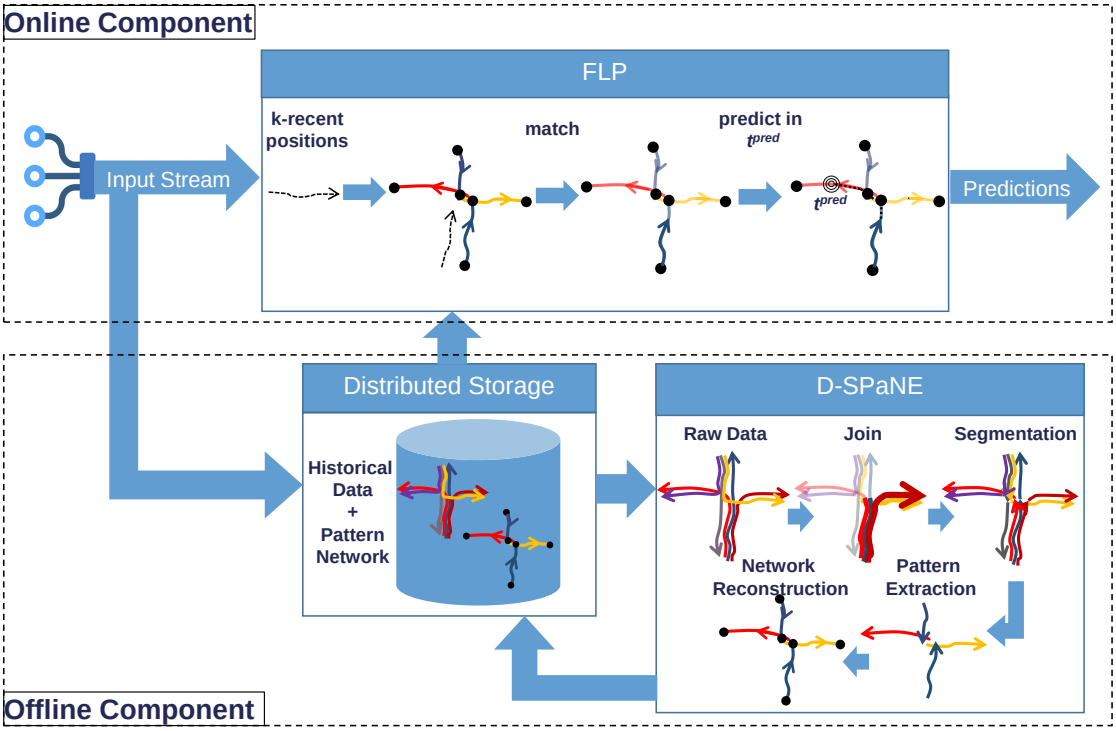

Figure 29: The PITHIA Architecture.

Definition 8 *(**Future Location Prediction**) Given a desired look-ahead time t^{pred}, a pattern network G and the recent k positions $\{r_{N-k+1}, \ldots, r_N\}$ of moving object r, where, r_N is the latest reported position, predict the position of r at t^{pred}, where $t^{pred} > r_N.t$.*

3.4.2.4 Problem Solution The PITHIA framework consists of an *offline* and an *online* component, as illustrated in Figure 29. The *offline* component is responsible for extracting the subtrajectory pattern networks in a distributed manner, given a large set of accumulated historical data. The *online* component receives streams of trajectory data and for each moving object, it retrieves its corresponding *hybrid* pattern network, "matches" its recent history on the network and routes through it until it finds the future location of the moving object at the desired look-ahead time t^{pred}.

Offline Component Concerning the *offline* component, it consists of a distributed storage file system, such as *HDFS*, which contains accumulated historical mobility data and *D-SPaNE*, which takes as input a distributed trajectory dataset from the distributed file system and constructs a set of *hybrid* subtrajectory pattern networks $SPN = \{SPN_1, \ldots, SPN_N\}$, where N is the number of moving objects. As already mentioned, the term *hybrid* indicates that we build a predictor for each moving object by taking into account its individual past movement when available and when not, collective historical patterns. An overview of *D-SPaNE* is presented in

Algorithm 4.

Algorithm 4 $D - SPaNE(D)$

1: **Input:** D
2: **Output:** set SPN of subtrajectory pattern networks
3: **Preprocessing:** *Align* and *Repartition* D;
4: **for each** partition $D_i \in \cup_{i=1}^{P} D_i$ **do**
5: **perform** *Point-level Join*;
6: **group by** *Trajectory*;
7: **for each** Trajectory $r \in D$ **do**
8: **perform** *Subtrajectory Join*;
9: **perform** *Trajectory Segmentation*;
10: **group by** D_i;
11: **perform** *Pattern Extraction* $\forall$ D_i;
12: **perform** *Refine Results*;
13: **group by** *Trajectory*;
14: **for each** Trajectory $r \in D$ **do**
15: **perform** *Network Reconstruction*
16: **return** SPN;

Initially, we temporally *Align* the first points of each trajectory starting at $t = 0$, in such a way that the temporal dimension depicts the duration since the start of a trajectory, and then *Repartition* the data into P equi-sized, temporally-sorted temporal partitions (files), which are going to be used as input for the join algorithm in order to perform the subtrajectory join in a distributed way (line 3). Note that this is actually a preprocessing step that only needs to take place once for each dataset D. However, it is essential as it enables load balancing, by addressing the issue of temporal skewness in the input data. Subsequently, for each partition $D_i \in \cup_{i=1}^{P} D_i$ and for each trajectory we discover parts of other trajectories that moved close enough in space and time (line 5). Successively, we group by trajectory in order to perform the subtrajectory join (line 8). At this phase, since our data is already grouped by trajectory, we also perform trajectory segmentation in order to split each trajectory to subtrajectories (line 9). In turn, we utilize the temporal partitions created during the *Repartition* phase and re-group the data by temporal partition. For each $D_i \in \cup_{i=1}^{P} D_i$ we perform the pattern extraction procedure (line 11). At this point we should mention that if a subtrajectory intersects the borders of multiple partitions, then it is replicated in all of them. This will result in having duplicate and possibly contradicting results. For this reason, as a final step, we treat this case by utilizing the *Refine Results* procedure (line 12). We should also mention that lines 5-12, will be executed twice, once for identifying *collective* and once for *individual* patterns. The actual difference between the two executions lies at the *Point-level Join* (line 5), where during the extraction of *collective* patterns, we discover for each trajectory parts of other trajectories that moved close enough in space and time that belong to different moving objects, while during the extraction of *individual* patterns, they need to belong to the same moving object. Finally, we perform the *Network Reconstruction* (line 15) procedure by grouping the identified *individual* patterns by

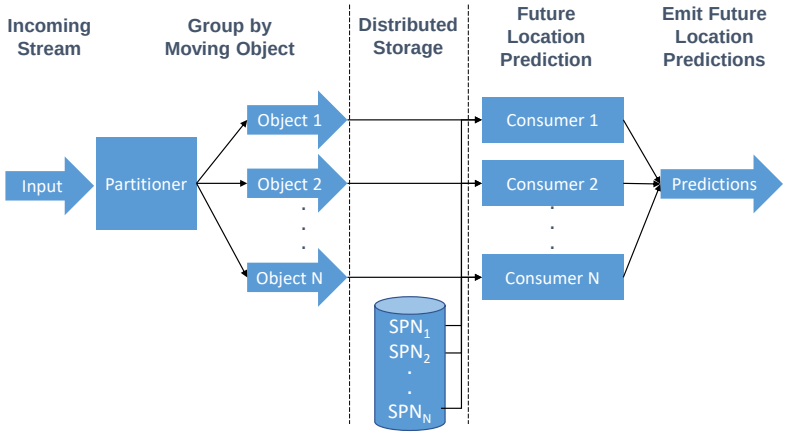

Figure 30: The *FLP* algorithm

trajectory, hence each processing node receives one such group, while we distribute the global patterns to all of the processing nodes. Finally, the set SPN is emitted.

Clearly, tackling the above problem is quite challenging in a distributed setting. For this reason, we outline a solution that follows the popular MapReduce paradigm.

Online Component Regarding the *online* component, it receives as input streams of mobility data D^{stream}, concerning the recent positions of moving objects and the goal is, for each moving object to retrieve its corresponding subtrajectory pattern network, match its k-most recent positions on the network and predict its future location at the given look-ahead time t^{pred}. Algorithm 5 describes and Figure 30 illustrates the online distributed future location prediction procedure.

Algorithm 5 $FLP(D^{stream}, SPN, t^{pred})$

1: **Input:** D^{stream}, SPN, t^{pred}
2: **Output:** set FLP of future location predictions
3: **partition** D^{stream} by moving object into $D_i^{stream} \in \cup_{i=1}^{N} D_i^{stream}$;
4: **for each** partition D_i^{stream} **do**
5: **retrieve** SPN_i;
6: $B_i \leftarrow k$-recent positions of D_i;
7: **match** B_i with the most-similar edge e_j;
8: **if** $Sim(B_i, e_j) > \alpha$ **then**
9: $FLP_i \leftarrow$ **predict** the future location of D_i^{stream} at t^{pred};
10: **return** FLP;

In more detail, the incoming stream of data is partitioned by moving object (line 3). Then, for each partition (i.e. moving object) D_i^{stream}, the corresponding subtrajectory pattern network SPN_i is retrieved. Subsequently, its k-recent positions are accumulated into B_i, which is a

actually a subtrajectory consisting of k timestamped positions, and the most similar edge e_j of SPN_i with B_i is identified lines 5-7). If their similarity is greater than a similarity threshold α, then the future location at t^{pred} gets identified by navigating through the network (8-9). Finally, the set FLP of future location predictions for all the moving objects is returned (line 10).

However, solving the above problem, in a distributed and streaming setting, is not trivial. For this reason, we utilize Big Data streaming technologies, such as the *Apache Kafka* Consumer interface.

3.4.2.5 Experimental Study In this section, we present the findings of our experimental evaluation. The experiments were conducted in a 49 node Hadoop 2.7.2 cluster. The master node consists of 8 CPU cores, 8 GB of RAM and 60 GB of HDD while each slave node is comprised of 4 CPU cores, 4 GB of RAM and 60 GB of HDD. Our configuration enables each slave node to launch 4 containers, thus up to 192 tasks (*Map* or *Reduce*) can be launched simultaneously. The offline component was implemented over Apache Hadoop and the online component by utilizing Apache Kafka for messaging and Kafka Consumers interface for stream processing.

For our experimental study, we employed a real, 1 week subset of the VFI dataset, consisting of 25019834 records.Furthermore, we utilized a synthetic dataset in order to verify that our solution operates as anticipated, given a dataset with a known ground truth. In more detail, **SMOD - Synthetic MOD (SMOD)** consists of an object which has performed 400 trips (trajectories) and is used for the ground truth verification. The scenario of the synthetic dataset is the following: the object moves upon a simple graph that consists of the following destination nodes (points) with coordinates: A(0,0), B(1,0), C(4,0) and D(2,1). Half of the times the object moves with normal speed (2 units per second) and another half moves with high speed (5 units per second). Figure 31 illustrates the 2D map of the SMOD consisting of three one-directional ($A \rightarrow B$, $B \rightarrow D$, $D \rightarrow C$) and one bi-directional road ($B \leftrightarrow C$). All objects move under the following scenario, for a lifetime of 100 seconds:

- (normal movement – 99% of the trajectories) All objects start from point A towards point B; the high-speed objects start at t = 0 sec and the normal-speed objects start at t = 20 sec. When an object arrives at B, it ends its trajectory with a probability of 15%; otherwise, it continues with the same speed to the next point. If there exist more than one option for the next point, it decides randomly about the next destination.
- (abnormal movement – 1% of the trajectories) A few outlier objects follow a random movement in space (other than these roads) with a speed that is updated randomly.

Our experimental methodology is as follows: Initially, we verify that our solution operates as anticipated by applying it to a dataset with a known ground truth. Subsequently, we utilize the real dataset and report the accuracy of the predictions and the performance, in terms of latency and throughput.

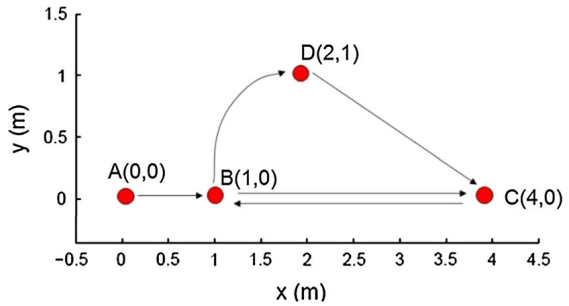

Figure 31: The 2-D map of SMOD

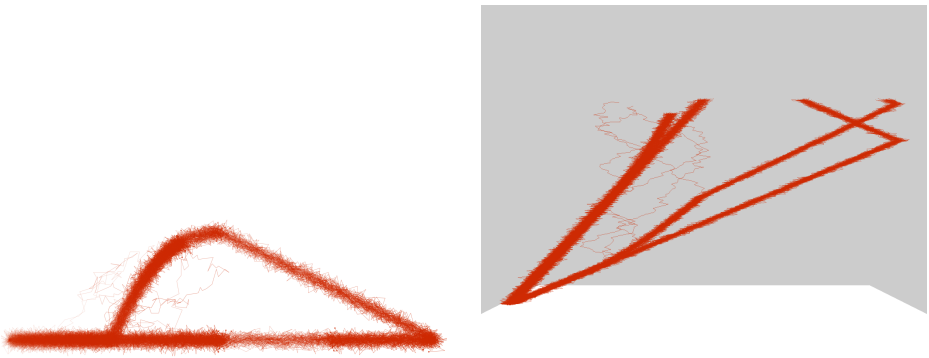

Figure 32: SMOD in the (a) xy-plane and (b) in 3D (the z dimension is time)

Ground truth verification In order to verify the correctness of our solution we utilized the aforementioned SMOD dataset. Figure 32(a) illustrates SMOD in the xy-plane and Figure 32(b) depicts the space-time cube of SMOD.

To begin with, we are going to verify that the pattern extraction component of our solution operates as anticipated. In more detail, the ground truth of the patterns that are hidden in SMOD can be inferred by the description of the dataset itself. In particular, eight clusters of subtrajectories need to be identified. Table 8 lists the eight clusters along with their spatial (2nd column) and temporal projection (3rd column).

Indeed, the pattern extraction module discovers these eight clusters, as illustrated in Figure 33,

The next step is the network reconstruction component, where the goal is to construct a directed graph G from the subtrajectory patterns. The challenge here is to restore the connectivity of movement of the specific objects by applying "stitches" when appropriate. In more detail, Figure 34 illustrates the edges (each pattern is an edge) of the graph before the graph reconstruction procedure and Figure 35 depicts the constructed graph, where the applied "stitches" are in black colour.

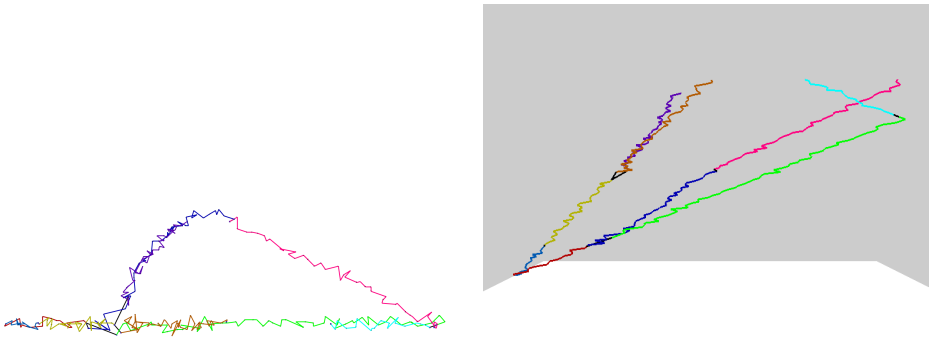

Figure 33: Discovered patterns in the (a) xy-plane and (b) in 3D (the z dimension is time)

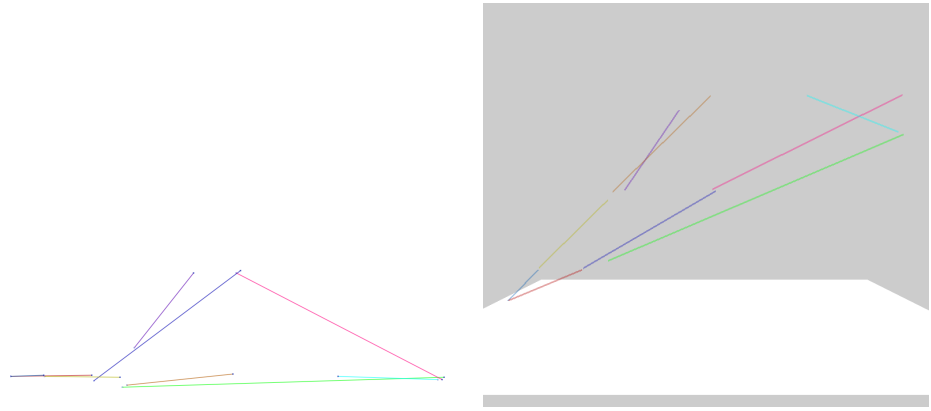

Figure 34: Network edges in the (a) xy-plane and (b) in 3D (the z dimension is time)

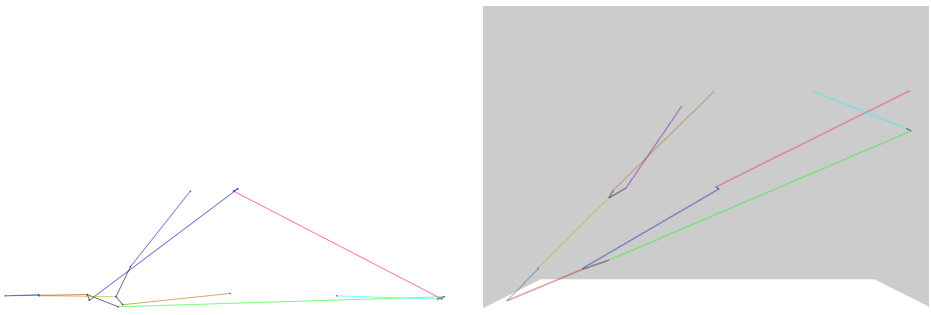

Figure 35: Reconstructed network in the (a) xy-plane and (b) in 3D (the z dimension is time)

Table 8: The ground truth hidden in SMOD

Cluster	Path	Time periods (clusters)
#1, #2	$A \rightarrow B$	[0, 0.2], [0.2, 0.7]
#3, #4	$B \rightarrow C$	[0.2, 0.8], [0.7, 1.2]
#5, #6	$B \rightarrow D$	[0.2, 0.52], [0.7, 1.2]
#7	$C \rightarrow B$	[0.8, 1]
#8	$D \rightarrow C$	[0.52, 1]

Quality of the predictions At this point we have verified that the offline component functions as expected. Regarding the accuracy of the predictions over the synthetic dataset we performed a set of experiments where we vary the look-ahead time and measure the Mean Average Error (MAE) of the predictions. In more detail, Figure 36 shows that the accuracy achieved is high, considering that the dataset diameter is approximately 500 meters and the look-ahead time ranges from 5%-30% of the dataset duration (each trajectory 'lives" for 100 seconds). Furthermore, as anticipated the larger the look-ahead time, the larger the MAE between the predictions and the actual positions.

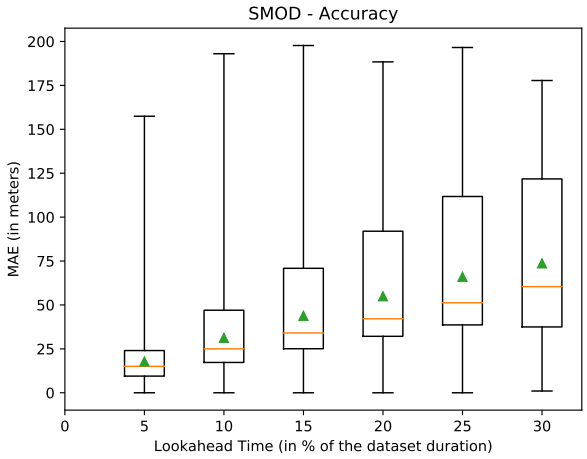

Figure 36: SMOD - Accuracy of the prediction in MAE

Regarding the VFI dataset, as depicted in Figure 37, the findings regarding the behaviour of MAE w.r.t. the look-ahead time are equivalent, the larger the MAE between the predictions and the actual positions.

Performance Finally, we investigate the performance of our solution in terms of latency (i.e. how much time it takes to make a prediction) and throughput (i.e. how many predictions

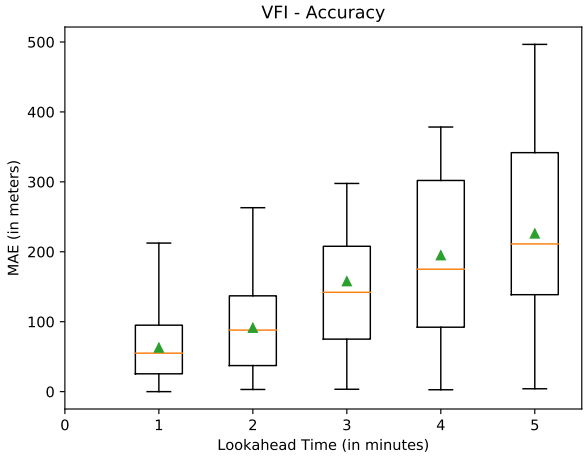

Figure 37: VFI - Accuracy of the prediction in MAE

per consumer our solution makes within a time unit). Initially, we utilized the SMOD dataset and measured the latency of our solution. Figure 38 presents our findings, where we can observe that our system, for the majority of the cases, can make a prediction at about 1 millisecond. In addition to that, we can see that the look-ahead time does not affect the processing time per prediction.

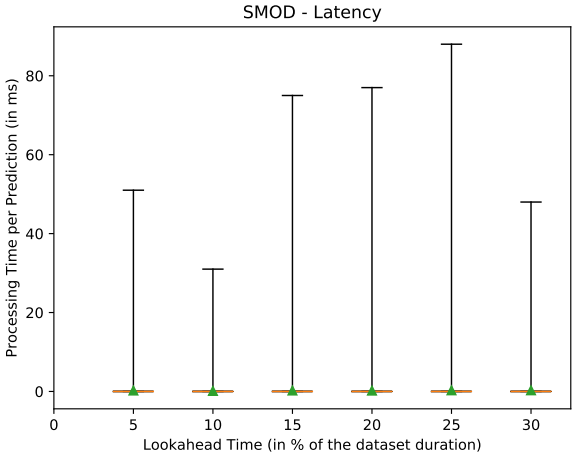

Figure 38: SMOD - Latency

Subsequently, we measure the throughput of our solution, in number of predictions per second. As illustrated in Figure 39, our system can make more than 500 predictions per second on average. Furthermore, we can also observe that that the throughput is not affected by the look-ahead time.

We run the same set of experiments, but this time for the VFI dataset. As expected, the

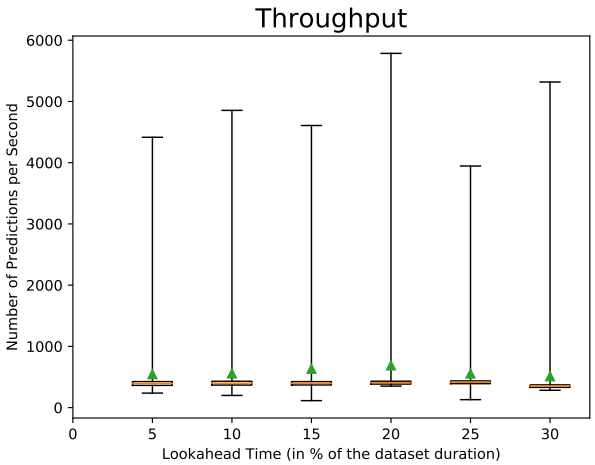

Figure 39: SMOD - Throughput

findings are again similar, which demonstrates the capability of our solution to handle large volumes of data in timely fashion. Figure 40 presents our findings, where we can observe that our system, for the majority of the cases, can make a prediction at about 1 millisecond.

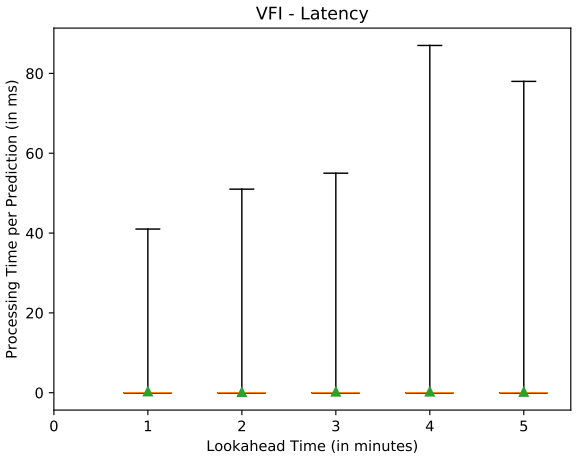

Figure 40: VFI - Latency

Finally, as illustrated in Figure 41, our system can make on average more than 500 predictions per second.

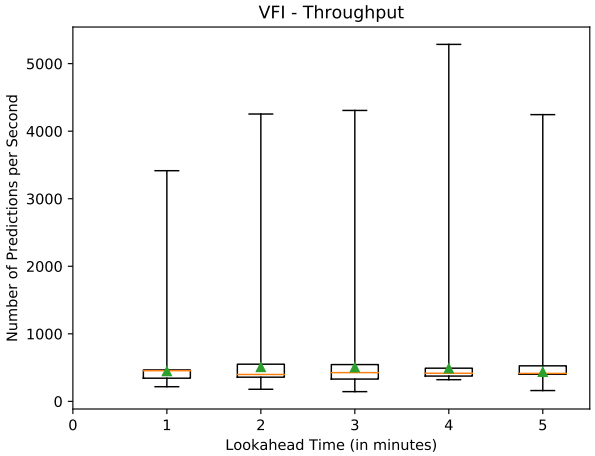

Figure 41: VFI - Throughput

3.5 Driver behavior profiling

In summary:

- *Generic question addressed*: Identify driving patterns and categorize driving behaviour on-the-fly based on tracjectory dynamics.
- *Track&Know specific question*: Driver Behaviour Profiling (DBP) in the short-term for online / offline trajectory analytics.
- *Novelty / Advantage over existing methods*: Completely unsupervised method (no ground truth required), using sparse GPS location data (no accelerometer) and context-aware enrichments (local speed limit), scalable complexity for the developed system (cascaded models)
- *Experiments conducted*: Experimental evaluation on a Track&Know Pilot dataset with quantitative and qualitative evaluations, comparison to current state-of-the-art unsupervised DBP methods.
- *Type of analytics*: Descriptive analytics.
- *Automation / TRL*: Currently being integrated for online processing mode for Pilot evaluation (TRL: 2 towards 4).
- *Extension to other domains*: Various types of vehicle mobility data with minimal data modalities available (sparse GPS, local speed limits), for fleet management, driver performance assessment, safety automation, etc.

Driver behaviour profiling, specifically in relation to identifying 'good' versus 'bad' drivers, is one of the most challenging problems in mobility data analytics. In this study [52], the core task of driver behaviour profiling is addressed at the minimum level of pre-requisites, i.e., GPS-only trajectory data (no accelerometer or other sensors) of low sampling rate (less than 0.1 Hz). A dynamic temporal resampling algorithm is employed for transforming GPS data into three distinct location-invariant time series, namely speed, acceleration, and turn rate, after map-matching and noise elimination pre-processing steps. A wide range of statistical, time series and spectral methods are implemented as feature functions or 'encoders' of various aspects of short-term mobility tracking. In our experimental study, a large real-world trajectory datasets are processed and transformed into feature-vector datasets, which are subsequently used in unsupervised training and adaptive category identification for the various driving behaviour 'states'. The proposed approach is designed for online/streaming mode and lightweight yet powerful analytics. The results show that such an approach is feasible, despite its challenging context of constraints described above, providing a data-driven adaptive way to recognizing 'normal' vs. 'abnormal' driving patterns on-the-fly.

3.5.1 Trajectory analytics for driver profiling

Trajectory analytics is one of the most commonly addressed tasks in the general context of geolocation data mining, usually involving mobility patterns, mobility graphs, points of interest (POI), hotspot detection, etc. A special topic that has been advancing steadily over the past few years is analysing the driving patterns and mobility dynamics as the driver 'behaviour', in the long- as well as in the short-term [115, 105]. In the case of long-term analytics, global trends and aggregated models can be discovered for regional and large-set statistics regarding driving habits, location- and route-specific risks of accidents, fuel consumption, delays due to traffic jams, POIs, etc. In the case of short-term, which today is the cornerstone in developing fully-autonomous driving vehicles [128], 'spot' analytics of the driving patterns within a limited time frame, usually no more than few minutes, provide hints about erratic driving, 'unpredictable' or risky movements, instantaneous violations of speed limits, etc. Both cases are very useful and challenging research problems, but their context, data modalities used and inherent methodological approaches are distinct and very different.

While the long-term approach in Driving Behaviour Profiling (DBP) has been explored using location-only data, the short-term approach is inherently more demanding in terms of spatio-temporal resolution, data quality and additional sensing modalities. In practice, tracking the movement of a single car or driver for an entire month to extract commonly used routes, visited POIs or risk of car crash within this context is inherently more straight-forward and well-studied than having to analyse movement patterns in the context of few minutes or seconds to distinguish between 'good' and 'bad' driving. The short-term case, being more challenging, is normally approached by employing multi-modal, high-resolution sensing, e.g. location tracking together with accelerometer measurements, while at the same time having pre-determined training routes

and confirmed driver 'events' as ground truth for model training [132, 188].

In general, short-term DBP is based on one or more of the following assumptions about the problem setup: (a) multi-modal sensing, typically location plus accelerometer, driver- or environment-sensing apparatus, etc; (b) high-resolution data especially in the temporal dimension, typically many samples per second; (c) specific annotation of 'good' and 'bad' driving patterns, either with some pre-determined set of driving 'events' that are introduced during test runs or by the labelling of training samples by a human expert [188]. In this work, the most challenging problem setup for the short-term DBP task is treated, i.e., when none of the previous assumptions is satisfied: neither (a) multi-modal sensing nor (b) high-resolution data nor (c) ground truth are available. This essentially translates into designing unsupervised predictive models for DBP [191, 16] in the short-term context when having location-only, sparse, variable-rate, unlabelled data. Each of these restrictions is a challenge by itself, given the fact that short-term DBP focuses on the fine details of movement patterns, which in this case have to be 'discovered' from data of very low quality and characterization for this purpose. However, these pre-requisites cannot always be satisfied, as multiple sensing and/or ground truth may not be unavailable, sampling rates may be too low, etc. Additionally, in this work the methodological approach and the proposed solution is developed in a way that leads to lightweight and on-the-fly processing, in order to be able to implement it as online/streaming service, which is essentially the true importance and value of having short-term DBP.

In summary, the novelties of this work in the DBP topic are the following:

- fully unsupervised, data-driven predictive models for DBP;
- use of sparse, variable-rate, GPS location data as input;
- online map-matching of the raw input to the road network & robust noise filtering;
- dynamic temporal resampling method for high-quality fixed-rate upsampling;
- treatment of three different data series: speed, acceleration, turn rate;
- extensive study on feature functions as 'encoders' of DBP patterns.

The rest of the paper is organized as follows: In Section 3.5.2 the short-term DBP, referred to simply as DBP from here on, is clearly defined in term of the modalities available, the definition of 'good' and 'bad' driving and the limitations posed by the current approaches; in Sections 3.5.3 through 3.5.7 the complete methodology of the proposed approach is described in detail, addressing each one of the individual challenges presented above; in Section 3.5.8 the datasets, experimental protocol and results are presented; in Section 3.5.9 the methodology is discussed in view of the presented results; finally, in Section 3.5.10 some conclusions are drawn for the proposed approach and its applicability to real-world DBP setups.

3.5.2 Problem description

Before the DBP problem is explored in detail in various aspects and limitations, a more formal definition of the context is required in relation to 'good' and 'bad' driving. Although there is no universal definition of the DBP problem, the most generic aspect that defines what is the core value at stake is *safety*, translated as not being causally involved in car accidents, i.e., not suffering from or causing them to others [160, 38].

In general, there are two sets of specifications or constraints that dictate if a driving behaviour is *safe* or not: (a) 'hard' limits that need to be strictly satisfied and (b) 'soft' restrictions that indicate some strong preference. In practice, (a) are regulations defined by laws and, thus, are almost always quantifiable and in some way inferred directly from data measurements, as for example *over-speeding* is a direct violation of the speed limit on a road. On the other hand, (b) can be an informal or qualitative description of safe driving for a single car and the others around it, as for example avoiding *cornering* (harsh turns or lateral movements), *harsh accelerating* or *harsh braking*, etc. [46]

In view of the relevant literature and the current state-of-the-art, in this work the DBP is treated in the context of two specific factors for road safety:

1. *Speed limits*: Driving patterns include checks against over-speeding conditions; these are hard limits that are available locally for each road, according to official regulations.

2. *Path predictability*: Driving patterns are associated to road safety at a lower or higher degree according to how predictable the trajectory of the car is; in other words, the more predictable a car's path is, the safer its driving profile is for everyone (anticipate and avoid the risk of accidents).

It should be noted that other criteria for 'good' and 'bad' driving may also be applied, including economic factors, environmental impact, time schedule, etc. However, safety is typically the single most important factor and the highest priority when viewing the DBP task in the short-term, e.g. when designing systems for fully autonomous driving [128].

3.5.2.1 Data availability and modalities Regarding data availability, DBP can be categorized according to the sensing modalities that are available for use:

- *single- or multi-vehicle*: Sensing data by/for individual cars, e.g. location or acceleration [167], versus being able to correlate or simultaneously track multiple cars close by, e.g. cars inside a buffer zone around it [72, 149].

- *without or with driver tracking*: Vehicle data may be supplemented with sensors that are tracking actual driver attributes, e.g. attention drift (eyes), sleepiness (steering wheel), etc. [17]

- *without or with environment tracking*: Vehicle data may be supplemented with sensors that are tracking external factors other than neighbouring cars, e.g. road lines/edges/signs, obstacles, etc. [189]

In real-world DBP applications, there are several limitations that may arise in relation to one or more of the aspects described above. The most common one is the lack of additional modalities other than location tracking and (maybe) accelerometer measurements, as these are readily available in off-the-shelf portable devices like typical smartphones [39, 26, 61, 75, 132]. In contrast, the other options usually require special devices installed inside the car (e.g. tracking cameras), around the car externally (e.g. proximity sensors, LiDAR), in combination with the other cars in traffic (e.g. inter-vehicle networking) or in combination with environmental guides (e.g. UV painting on road edges/signs). For obvious reasons, the cheapest and most preferable DBP solution would require location-only data, perhaps accompanied with acceleration measurements from sensors, if available.

3.5.2.2 Novelties of the proposed approach As previously described, the context of DBP can be very restrictive in terms of data availability and quality, sensor modalities employed and the existence of a reliable baseline to be used as ground truth for the models. Moreover, the type and complexity of the required processing can be prohibitive for on-the-fly DBP models that need to work with new data as they are generated, instead of processing them offline in batches with little or no processing time restrictions.

This work presents a new approach to DBP in the short-term context and with on-the-fly processing in mind. More specifically, the main focus and contributions in this work are the following:

- data-driven, purely unsupervised model training, without any labelled ground truth available;
- dynamic temporal resampling method for high-quality fixed-rate upsampling;
- application of high-quality map-matching to the underlying road network and robust noise filtering (pre-/post-processing);
- use of sparse (< 0.1 Hz), GPS location data of variable sampling rates for the single-vehicle DBP task;
- generation of high-quality multi-modal time series from the GPS data (speed, acceleration, turn rate);
- instead of simple thresholds, in-depth analysis of the data series with optimally selected higher-order ('texture'), curve and spectral features;
- association with external data enrichments, e.g. weather and road/vehicle types, as additional DBP features;

- employment of multi-stage clustering as 'blind' DBP state tracking, i.e., driving 'categories' that are discovered naturally from the data;
- employment of low-complexity, on-the-fly processing, to enable DBP applications for on-line/streaming modes.

In order to stay at the minimum level of pre-requisites, in our work we adopt a dynamic temporal resampling method that is employed for transforming the sparse GPS-only trajectory data into three distinct, optimally upsampled to a fixed-rate and location-invariant time series, namely speed, acceleration and turn rate. Additionally, the feature functions are optimally selected for analysing these reconstructed data series as content-rich 'encoders' of DBP patterns but yet lightweight enough to be applicable to on-the-fly processing architectures.

Given the data restrictions and the challenging setup of the DBP problem here, only few works from the current best-practices in DBP are comparable with this proposed approach [160, 105, 115, 110]. The most 'compatible' work in terms of unsupervised DBP categorization via clustering context-sensitive (per road segment) speed and acceleration descriptive statistics is [191]; this method is also implemented and included in the experimental work here for providing comparative results in the same dataset, as described in Section 3.5.8.

The results show that our approach is very efficient and computationally feasible even for on-the-fly processing, thus providing a data-driven adaptive way to recognize various driving patterns.

The overall 'pipeline' view of the proposed approach can be summarized in the following sequence of phases:

1. map-matching & filtering of the raw GPS data;
2. Dynamic Temporal Resampling Buffer (DTRB);
3. feature extraction via trajectory analytics;
4. feature selection for dimensionality reduction;
5. DBP evaluation via unsupervised models (clustering).

The following sections describe the methods developed and applied in each phase.

3.5.3 Road matching and filtering

As described earlier, instead of using the raw GPS data, the location points are map-matched against the underlying road network for removing GPS uncertainty and some of the noise. Map-matching is a well-studied problem in mobility data management and analytics [21, 107] . Nevertheless, in our approach, the distances of each point from the nearest road segments are estimated geometrically using the Haversine function (spherical approximation) and they are used as input to an online map-matching module that is based on Hidden Markov Model [58]. This enables

the correction of GPS errors not just for single points but for entire sequences along the 'most probable' path in the maximum-likelihood sense. Furthermore, in this work the core HMM-based map-matching process has been augmented with an additional step of localized pre-fetching and thresholding of the underlying OpenStreetMap (OSM) road network, in order to speed up the process and avoid singular road matches at excessive distances, i.e., discard the point instead as GPS noise.

Given the map-matched GPS trajectory, with some outlier points already removed as noise, the 'most probable' path is examined for any 'spot' violations against a set of predefined thresholds relevant to the expected values for distance versus time step. Validity checks can be asserted as additional post-processing for realistic maximum speed, acceleration, braking and turn rates, hence any location points resulting in such violations are also removed as GPS noise. The end result from this entire process is the maximum-likelihood 'corrected' trajectory of the low-resolution GPS track, which is subsequently used as input for the next steps of the proposed method. Figure 42 illustrates a close-view comparison of the raw GPS location data (in red) and the map-matched & noise-filtered trajectory (in blue), as already discussed above.

3.5.4 Dynamic Temporal Resampling Buffer (DTRB)

The constraint of having sparse GPS location data and no other modality available is one of the most demanding challenges addressed in this work, since DBP in the short-term context requires high-resolution movement analytics, i.e., detection of over-speed at one moment not on averaged values, 'spikes' in the acceleration or turn rate, etc.

The core idea of the DTRB is that the map-matched & filtered low-resolution GPS track is analysed for detecting sequences where the sampling rate is adequately high, even for short periods of time or data 'slices'. A much higher and fixed sampling rate is applied to the source data series for upsampling. This is implemented using high-quality shape-preserving interpolation in the entire data slice, specifically a spline-like piecewise cubic Hermite interpolating polynomial ('pchip') [49, 89]. For each such slice of sparse GPS location data, the most recent part (temporally) of its upsampled transformation is used as the basis for producing the three main data series used here for DBP, i.e., speed, acceleration and turn rate. These are used as input to the feature extractors in the next step, which essentially detect, encode and quantify the properties that are relevant to the DBP task.

The 'pipeline' outline of the DTRB algorithm can be described as follows:

- continuously scan the incoming location data for 'dense' slices;
- when a valid slice is detected, upsample to a fixed rate;
- from the upsampled location data, generate speed, acceleration, turn rate series;
- perform another set of validity checks for the generated data series (filtering);
- forward the processed (3x) data series for feature vector generation.

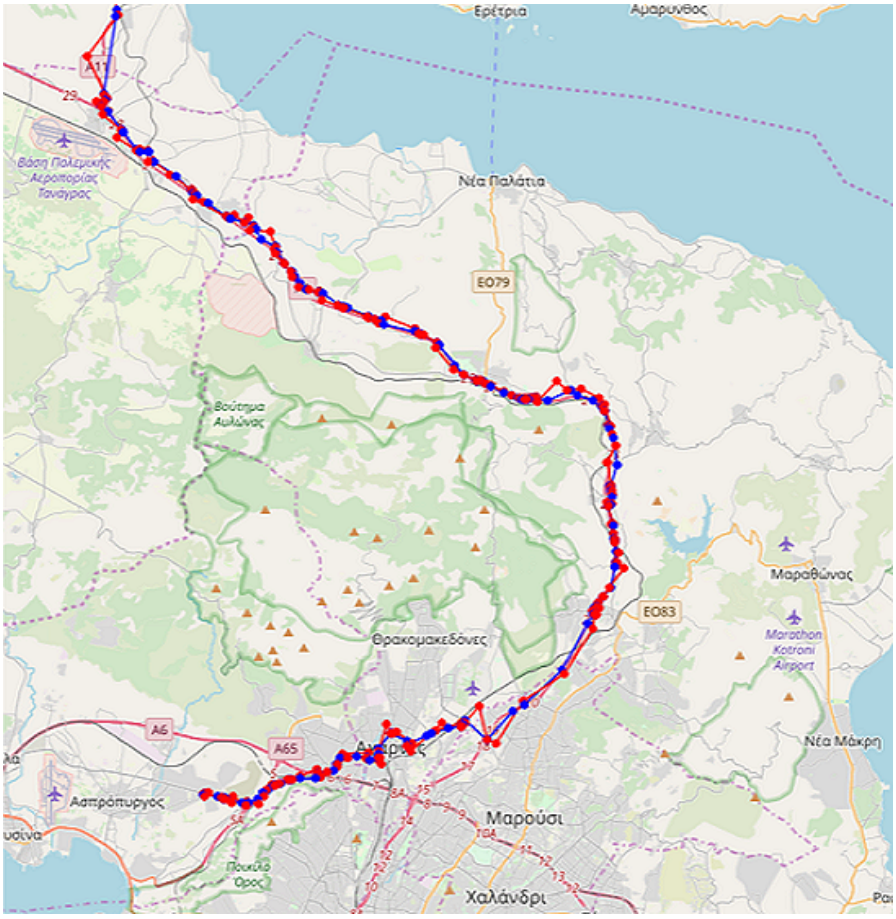

Figure 42: Example of raw GPS data map-matching & filtering from the dataset used.

Taking all the design constraints into account, as well as the need to continuously process the input on-the-fly as new GPS location data arrive, the DTRB algorithm satisfies the following requirements, re-checked upon every new input:

1. *Input*: Wait for new GPS location points (or read next from offline file).
2. *Spatial span* (check): Current slice contains at least N_s^{min} GPS location points.
3. *Temporal span* (check): Current slice spans at least L_s^{lim} sec.
4. *Temporal inter-distances* (check): Between L_s^{min} and L_s^{max}.
5. *Detect gaps*: Whenever L_s^{max} is violated, gap is detected and the sequence buffer is flushed, keeping only the current location point.
6. *Short slices*: If L_s^{min} is satisfied **and** $N_s = N_s^{max} > N_s^{min}$ location points are available, consider the slice as valid even when its total time span $L_s < L_s^{lim}$.
7. *Density criterion*: As soon as new input results in $N_s \geq N_s^{min}$ **and** $L_s \geq L_s^{lim}$ **and** L_s^{min}, L_s^{max} are satisfied, the slice is marked as valid and is forwarded for further processing; otherwise return and continue from (1).
8. *Upsampling*: For every valid slice detected, produce speed U_t, acceleration A_t and turn rate R_t data series from the GPS location points, upsampled at fixed rate T_s.
9. *Validation*: For each of the three new data series, perform a set of additional range checks[3]; discard the slice if any check is invalidated and return to (1).
10. *Output*: If all checks validate ok, use the most recent $n \cdot T_s$ part of the upsampled data series U_t, A_t, R_t for DBP feature vector generation.

The DTRB configuration parameters can be tuned according to the specific dataset at hand. Figure 43 presents a simplified example of DTRB functionality in various data input conditions, while Figure 44 illustrates the internal DTRB pipeline for the post-processing validity checks, i.e., between upsampling and feature vector generation (see steps 8-9-10 above).

3.5.5 Feature extraction via trajectory analytics

According to the definition of the DBP task as described in Section 3.5.2, it is clear that, at least for the short-term context, using simple thresholding in relation to fixed limits, e.g. checking for over-speeding or against the mean value of speed when traversing a specific road, is very inefficient. Instantaneous violations can be missed when the sampling rate of data is too low or when averaged over a temporal frame that is too large. Most importantly, these threshold-based methods using simple 1st-order descriptive statistics, e.g. mean value or standard deviation

[3]Range checks: $0 \leq U_t \leq 55.50$ m/sec (200 km/h); $A_t \leq 10.29$ m/sec^2 (0-100 km/h in 2.7 sec); $R_t \leq 90$ deg/sec (1.5708 rad/sec).

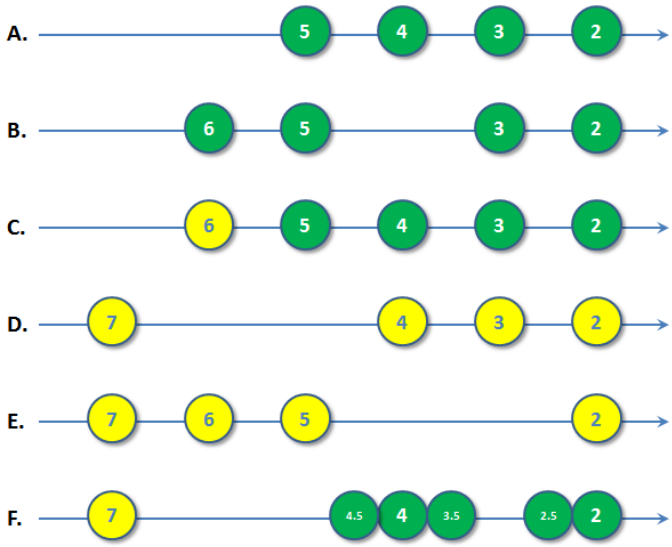

Figure 43: Simplified example of DTRB functionality. Each node represents a data point (input) and the number inside is the $|dt|$ from the current time $t = 0$ sec. DTRB configuration is: $N_s^{min} = 4$, $N_s^{max} = 5$, $L_s^{min} = 0.5$, $L_s^{lim} = 3$, $L_s^{max} = 2$. Green (dark) nodes are valid slices for further processing, while yellow (light) nodes are not.

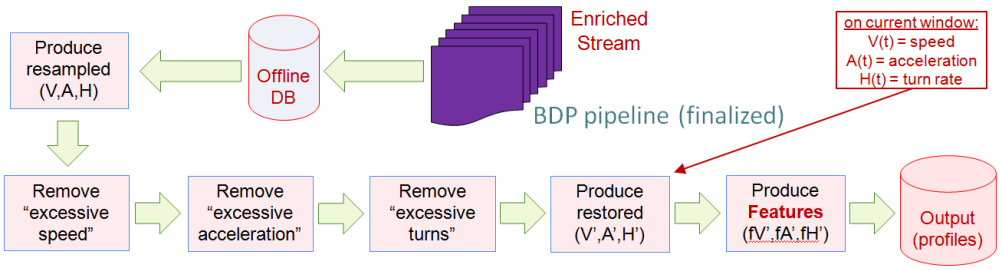

Figure 44: DTRB internal pipeline for validity checks.

of speed, max value of acceleration, etc., are not adequate in capturing the actual fine-scale properties of the trajectory, as required for truly effective and robust DBP.

In this work, a very large set of candidate feature functions is employed as the initial pool of DBP trajectory analytics, ranging from 1st- and 2nd-order descriptive statistics to curve, spectral and synthetic features. In summary, this initial feature set includes:

- *1st-order statistics*: min, max, (arithmetic) mean, median, mode, stdev, range, skewness, kurtosis, entropy, geometric mean.
- *Curve statistics*: zero-crossings, roughness index, correlation vs. time, linear regression coefficients, curve vs. geometric length.
- *Synthetic*: ratios between selected 1st-order statistics, e.g. range vs. stdev.
- *Spectral*: auto-regressive AR(2) coefficients, signal 'energy'.
- *2nd-order statistics*: Haralick features [106], run-length features [175].
- *Enrichments*: vehicle type, road type, road speed limit.

In the current state-of-the-art in DBP, most works exploit features from the 1st-order statistics category, mostly because they are easy and fast to calculate and straight-forward to interpret [105, 115, 191]. Some of the curve statistics are also easy to calculate, but usually less effective or significantly correlated to other 1st-order statistics, e.g. zero-crossings with standard deviation. To the best of our knowledge, most of these feature functions have not been used in this context of DBP, i.e., having only sparse, variable-rate, GPS location data as input. The rationale for the groups of features described above is that they capture an information-rich and 'compressed' form of the trajectory properties that are directly or indirectly related to the DBP task at hand.

The entire set of the initial pool of 45 feature functions listed above is applied separately for each of the three data series, i.e., speed, acceleration and turn rate. In addition, the feature set is supplemented with several enrichments (e.g. GPS quality), from which three are DBP-related: vehicle type, road type and road speed limit. As such, the final feature vector, generated for each valid slice produced by DTRB, contains 45 feature functions for the three data series plus three more from enrichments, i.e., a total of $45 \cdot 3 + 3 = 138$ features or 'encoders' of potential DBP mobility patterns in the short-term context. Also, since all the data restrictions of the DBP task have already been addressed by the DTRB (sparsity, noise, road map-matching, upsampling), this feature generation stage is independent and it can be applied in principle to any other DBP setup. Figure 45 illustrates some of the processing implemented for translating a very small set of GPS reference points into upsampled data series and feature values extracted from it.

Since in this study the DBP task is addressed in its fully unsupervised mode, the goal is to identify 'interesting' features that exhibit explicit statistical characteristics, for example multi-nomial distributions and/or heavy tails, in order to produce clear data groupings and/or extreme zones, respectively. In turn, these can produce more evident and distinct clusters or 'categories' of

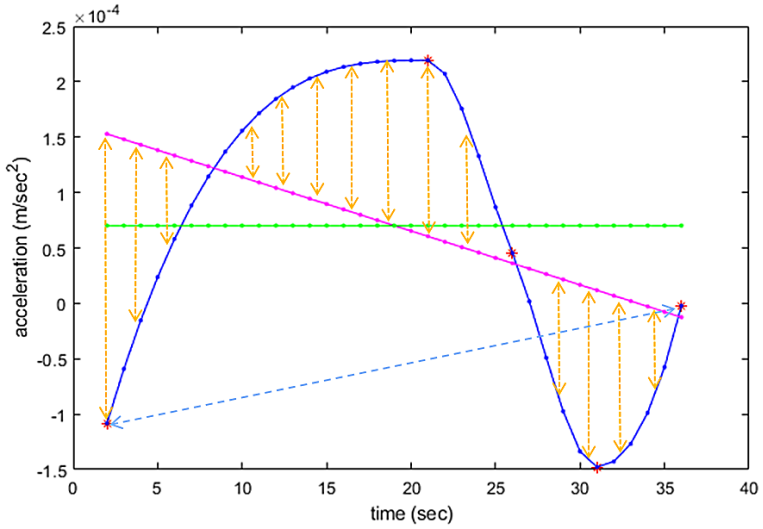

Figure 45: Example of DTRB processing for transforming a low-resolution variable-rate data 'slice' (acceleration) into an upsampled fixed-rate version and modelling for feature extraction; blue is the resampled curve length, magenta is the linear regression trend, green is the mean value, yellow is the signal energy.

driving behaviours. In other words, the more explicit these characteristics are, the easier it is for unsupervised models to be trained for detecting 'normal' versus 'abnormal' driving, as described later on in Section 3.5.7. Figure 46 illustrates an example of a feature with low information content for this task, i.e., very narrow Gaussian distribution with very low skewness (no heavy left/right tails). On the other hand, Figures 47 and 48are examples of such information-rich 'encoders' of clear and distinct groupings of DBP patterns - these are actually the three best-ranked features selected at the end of the dimensionality reduction process, as described next in Section 3.5.6.

3.5.6 Feature selection for dimensionality reduction

The initial set of feature functions employed is 45 for each data series, i.e., speed, acceleration and turn rate, plus three more included from data enrichments (vehicle type, road type, road speed limit), thus resulting in a total of 138, as described in Section 3.5.5. This collection of candidate 'encoders' of DBP patterns includes essentially various categories of time series analytics, statistics, signal processing and image analysis algorithms, adapted here for 1-D data series. Before any model training, the features set has to be refined and significantly reduced in size, in order to decrease the dimensionality of the DBP feature vectors dataset and, thus, the complexity of the models.

The feature selection and dimensionality reduction process applied here consists of a multi-step approach, incorporating statistical ranking, factor analysis, predictive model evaluation, etc.

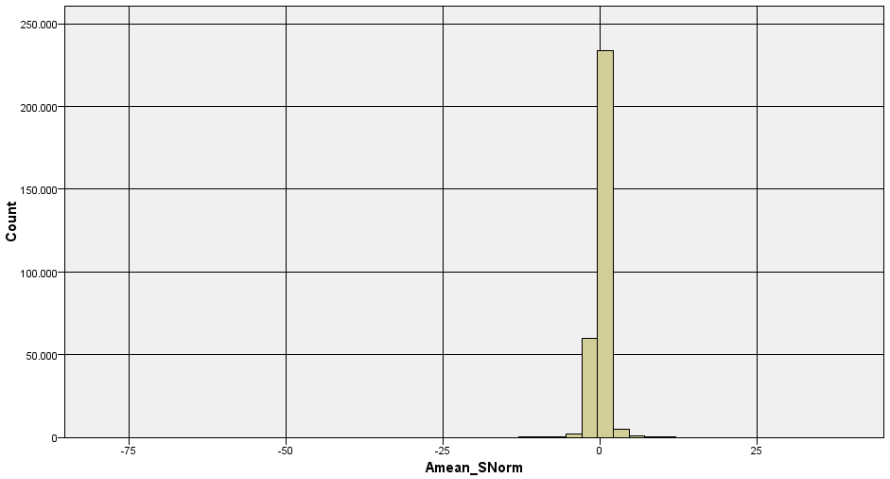

Figure 46: Example of 'bad' (information-poor) feature function for DBP (acceleration: A_t^{mean}).

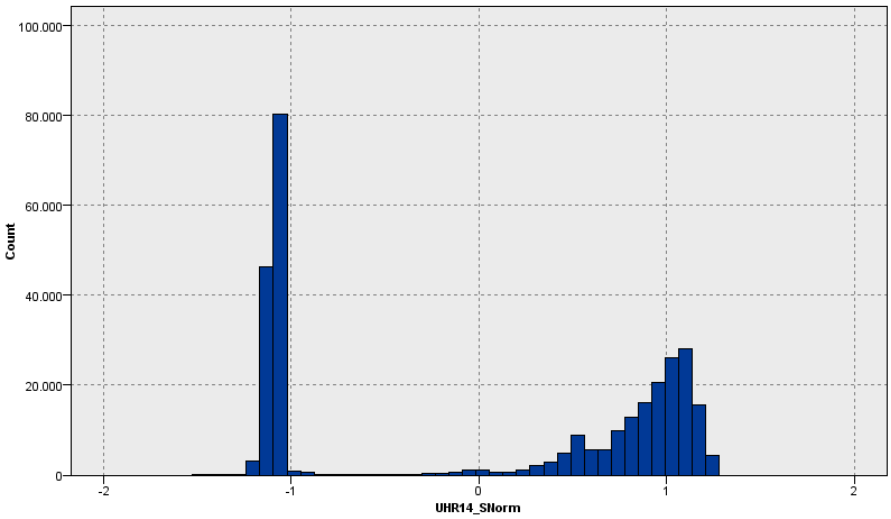

Figure 47: Example of 'good' (information-rich) feature function for DBP (speed: U_t^{HR14}).

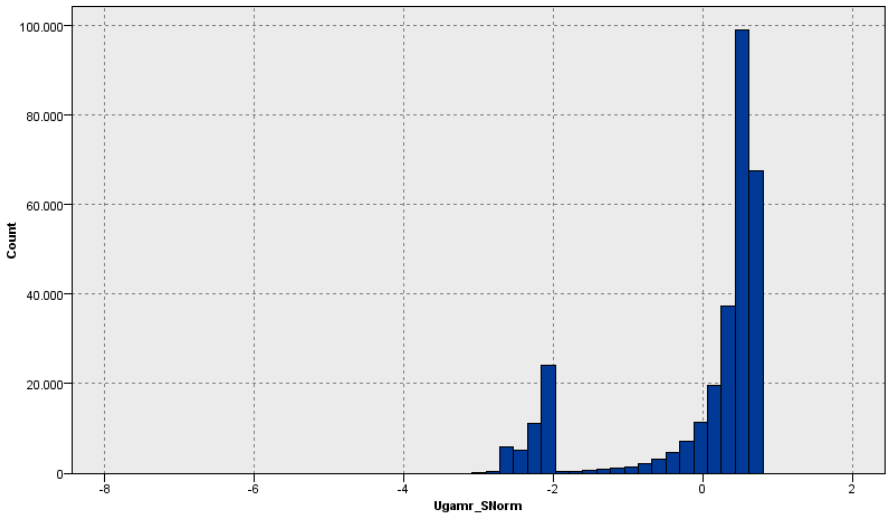

Figure 48: Example of 'good' (information-rich) feature function for DBP (speed: U_t^{gamr}).

More specifically, the first stage is comprised of the following:

1. *Single-variate analysis* (SVA): Entropy, kurtosis, quartiles, standard deviation.
2. *Limits-based analysis* (LVA): Adaptive labelling & hypothesis testing against outlier/extreme zones.
3. *Goodness-of-Fit analysis* (GoF): Kolmogorov-Smirnov test, Jarque-Bera test, Lilliefors test.
4. *Multi-variate analysis* (MVA): Pairwise correlation, mutual information, cross-entropy.
5. *Factor analysis* (PCA): Principal Component Analysis for ranking based on eigenvectors.
6. *Fractal dataset analysis* (FDA): Intrinsic dataset dimensionality analysis.

Next, feature selection via model testing is employed using the refined subset of 31 features, a union of the top-10 best-ranked features from each selection method, in order to investigate and identify even smaller features subsets still capturing most of the DBP information. More specifically, a Expectation–Maximization (EM) algorithm for fast clustering was employed with several options for heuristic features subset evaluation, including forward-backward selection, randomized subsets, particle swarm optimization (PSO), genetic/evolutionary programming, etc. Clustering quality metrics, more specifically Silhouette and Fisher criterion [178], were employed as quality metrics for the resulting clusterings and, thus, a ranking method for each setup. Additionally, due to the significantly reduced size of the features subset, visual analytics were also employed for identifying 'interesting' features w.r.t. their PDFs, i.e., multi-nomial, heavy-tailed, etc.

Finally, a third stage of feature refinement produces further shrinkage of the dimensionality is achieved, from 31 down to three main features, plus a supplementary subset of eight top-ranked features that are combined into PCA components, as described next in Section 3.5.7.

3.5.7 Unsupervised learning - Clustering

Using the refined features subset from the iterative selection process described in Section 3.5.6, the design of the unsupervised models includes clustering. More specifically, K-Means with Euclidean distance and speed & acceleration statistics as input was implemented as a reference baseline of the most comparable state-of-the-art approach in the relevant DBP literature [191]. Additionally, two-step or BIRCH clustering with log-likelihood as distance function is employed in this study [199] as the base unsupervised model.

There are two main reasons why BIRCH clustering is selected as the main algorithm here instead of K-Means. First, it incorporates a pre-clustering step that enables the automatic selection of k for the number of clusters. Second, it incorporates a log-likelihood function as distance metric instead of the Euclidean distance in standard K-Means, hence it is more distribution-agnostic. In practice, this means that the underlying probability distribution for each dimension is not assumed as strictly Gaussian and, hence, the cluster boundaries are more well-fitted to the actual training data. This is verified in the experimental part of this work, where in very similar clustering setups with K-Means and BIRCH algorithms, the second one produced cluster boundaries that are more orthogonal against each axis (input dimension), i.e., a model more easily implementable via optimal thresholding per-feature instead of minimum-distance calculations against the centroids in the combined feature space.

Furthermore, a multi-stage approach is employed as a composite clustering model, with each level incorporating a separate BIRCH clustering model using only specific features from the input. The optimal selection of features in each case is part of the model design in each clustering level. Again, Silhouette (mostly) and Fisher criterion are employed as quality metrics for the resulting clusterings and a quantitative ranking method for each setup, as well as some qualitative assessment by visual analytics and inspection.

In summary, the following clustering levels are trained in a cascaded form:

- Level-1 (**TSL1**): BIRCH clustering using U_t^{HR14} and U_t^{gamr} as input, resulting in 4 clusters as output.
- Level-2 (**TLS2**): BIRCH clustering using TSL1 cluster id and U_t^{spen} as input, resulting in 8 clusters.
- Level-3 (**TSL3**): BIRCH clustering using TSL2 cluster id and 2 PCA factors as input, resulting in 5 clusters.

In practice, TSL1 uses only two features from the speed data series, namely U_t^{HR14} and U_t^{gamr} described below, to produce the first level of clustering; as their corresponding PDFs

illustrate in Figures 47 and 48, these two features effectively produce a very clear four-cluster setup. Similarly, using the output from TSL1 and U_t^{pen} as additional input, another two-category input effectively produces a very clear eight-cluster output in total, as described in detail later on in Section 3.5.8. Finally, the additional clustering TSL3 can be incorporated for even finer and in-depth analysis of the DBP features, if the processing complexity of PCA is acceptable for the application at hand. In this case, the input space is PCA-transformed and, thus, neither the input or the output dimensionality is directly comparable to the ones employed in TSL1 and TSL2 models.

Some of the feature functions used here are based on 2nd-order statistics or 'texture' of a data series, more specifically the well-studied set of 14 features that use the Co-Occurrence Matrix (COM) [106] and 6 features that use the Run-Length Matrix (RLM) [175]. COM is defined as a Np-by-Np matrix $p(i,j)$ that counts pairs of subsequent discrete values $\{i,j\} = \{1,\ldots,Np\}$, where Np is the number of bins used to discretize the continuous range of the target variable, i.e., the same per-series data ranges used for the validity checks in DTRB (see Section 3.5.4). Similarly, RLM is defined as a Np-by-Nr matrix $r(i,j)$ that counts same subsequent discrete values $i = \{1,\ldots,Np\}$ of sequence lengths or 'runs' $j = \{1,\ldots,Nr\}$, where Np is the number of bins used to discretize the continuous range of the target variable and Nr is the maximum expected 'run'. Whenever the defined Np and Nr discretization lowest or highest limits are exceeded, the corresponding marginal bins are used for the counter updates, i.e., value inside or higher/lower than the discretization limits.

The features used in the TSL1 and TSL2 models are the following:

- *Maximum Correlation Coefficient*: (speed)

$$U_t^{HR14} = \sqrt{\lambda_2} \tag{8}$$

  where λ_2 is the second-largest eigenvalue of:

$$Q(i,j) = \sum_k \frac{p(i,k)p(j,k)}{p_j(i)p_i(k)} \tag{9}$$

  and: $k = \{1,\ldots,Np\}$, $p_j(i) = \sum_{j=1}^{Np} p(i,j)$,
  $p_i(j) = \sum_{i=1}^{Np} p(i,j)$.

- *Geometric-to-arithmetic means ratio*: (speed)

$$U_t^{gamr} = \frac{\sqrt[N]{\prod_{i=1}^{N} u_i}}{1/N \sum_{i=1}^{N} u_i} \tag{10}$$

- *Road speed penalty factor*: (speed)

$$U_t^{spen} = sign(U_t^{vpen} - 0.98) \tag{11}$$

where:

$$U_t^{vpen} = \frac{U_{rlim}}{\max(0, U_t - U_{rlim}) + U_{rlim}} \tag{12}$$

Note that, since all calculations are applied to discrete- rather than continuous-valued series for U_t, the term u_i in Eq.10 is essentially identical to U_t for $i = t$.

Regarding the penalty factor related to the (local) road speed limit, it is $0 < U_t^{vpen} \leq 1$; in reality, in most cases $0.7 \leq U_t^{vpen} \leq 1$. When $U_t \leq U_{rlim}$ then $U_t^{vpen} = 1$, i.e., speed strictly within the permitted limit, and when $U_t > U_{rlim}$ then $U_t^{vpen} < 1$, i.e., $U_t^{vpen} = \frac{U_{rlim}}{U_t}$. Thus, the threshold $U_t^{vpen} \lesseqgtr 0.98$ is translated to actual speed $U_t = U_{rlim}/0.98 \approx 1.02 \cdot U_{rlim}$ or speed at least 2% over the permitted limit. This is a natural, data-driven definition of 'safe' and 'unsafe' speed zones, given that in large avenues and highways the average speed of vehicles under normal traffic conditions is indeed very close to the actual road limit. Here, a hard-thresholded value U_t^{spen} at the level 0.98 is used instead of U_t^{vpen}. This is valid in the sense that, as described in section 3.5.2, 'hard' regulations e.g. for speed limits are one of the factors that define the DBP problem.

The features used in the TSL3 model are the first two PCA factors calculated for the subset of the following eight supplementary features:

- *Value non-uniformity*: (speed, acceleration, turn rate)

$$X_t^{RL03} = \frac{\sum_{i=1}^{Nx}(\sum_{j=1}^{Nr} r(i,j))^2}{\sum_{i=1}^{Nx}\sum_{j=1}^{Nr} r(i,j)} \tag{13}$$

where:

$$X_t = \{U_t, A_t, R_t\} \tag{14}$$

- *Sum entropy*: (speed)

$$U_t^{HR08} = \sum_{i=2}^{2N} p_{x+y}(i) \log p_{x+y}(i) \tag{15}$$

- *Mode-to-mean ratio*: (speed)

$$U_t^{gamr} = \frac{mode(U_t)}{1/N \sum_{i=1}^{N} x_i} \tag{16}$$

where $mode(U_t)$ is the value where the peak of the PDF occurs.

- *Range-to-stdev ratio*: (turn rate)

$$R_t^{rsr} = \frac{\max R_t - \min R_t}{\sqrt{\sum_{i=1}^{N}(x_i - \mu_x)^2/N}} \tag{17}$$

where $\mu_x = 1/N \sum_{i=1}^{N} x_i$ is the mean value of R_t.

- *AR(2) 1st-order coefficient* : (speed, acceleration)

$$X_t^{ar1} = \alpha_1 \ , \quad A(z)X_t = e_t \tag{18}$$

where $A(z) = 1 - \alpha_1 z^{-1} - \alpha_2 z^{-2}$ an AR(2) auto-regressive model of order 2 for the best-approximation (minimum error e_t) model identification of series X_t via the Yule-Walker algorithm [148] and $X_t = \{U_t, A_t\}$.

Based in this multi-stage clustering approach and the specific TSLx models designed for each stage, Section 3.5.8 describes the experimental protocol and the results for their assessment, using a real-world dataset as described next.

3.5.8 Experiments and Results

3.5.8.1 Datasets used In this work, an extensive real-world trajectory dataset of GPS location data was used as the basis, consisting of 977,646 records generated by special-purpose devices installed in 2,638 large vehicles (transport trucks) travelling in the main urban area of the city of Athens (Attica region, Greece) for a period of 24 hours in a typical weekday. More specifically, the dataset was defined within the spatial bounding box: Lat = [37.8860, 38.1057] North / Lon = [23.5591, 23.9128] East and temporal frame: 2-Nov-2018 (00:00':00"-23:59':59"). The data enrichments included various parameters regarding the local weather (precipitation, temperature, wind speed & direction from NOAA), the underlying road network (OpenStreetMap – OSM) and the GPS signal quality (number of satellites tracked, map-matching distance & probability as described next).

3.5.8.2 Experimental work & results The experimental protocol in this study was based on the real-world dataset as described above.Most of the experimental work followed the five-phase sequence summarized at the end of Section 3.5.2, while some parts required iterations between feature subset refinement and model design for clustering (see Sections 3.5.6 and 3.5.7, respectively). Various hardware/OS[4] and software[5] platforms were used for the experimental work, some of which is currently ported to R, Java and Python for open cross-platform use.

For DTRB, Figure 49 illustrates the 3-D histogram of the number of extracted valid slices from the data per data points included and per temporal span used, which was the main guideline for the optimal configuration of the DTRB parameters for the dataset at hand. More specifically, considering all these constraints and after extensive experimentation with the DTRB configuration, the nominal process is defined as having a slice of least $N_s^{min} = 4$ location data points within a span of $L_s^{lim} = 32$ sec, being at least $L_s^{min} = 1$ and no more than $L_s^{max} = 32$ sec apart. If $N_s = N_s^{max} = 6$ location points and L_s^{min} is satisfied, then the slice is considered

[4]Intel core i7-3537U @2.00GHz & 8GB memory; Intel core i7-8550U @1.80GHz & 32GB memory; Microsoft Windows 8.1 & 10; Ubuntu Linux 19.04 & 18.4 LTS.

[5]Mathworks MATLAB v9.4/R2018a (x64); Octave v5.1.0; R v3.6.2; WEKA v3.9.4; IBM SPSS Modeler v14.1 & Statistics v26; custom Java & C/C++ tools for data import/export.

as valid regardless of L_s. This means that the total temporal extent of the slice may be from $N_s^{max} \cdot L_s^{min} = 6 \cdot 1 = 6$ up to $N_s^{min} \cdot L_s^{max} = 4 \cdot 32 = 128$ sec. The fixed rate for upsampling was set to $T_s = 1$ sec and the most recent $n \cdot T_s = 32 \cdot 1 = 32$ sec is the span of the most recent part of the slice that is produced as output as Figure 49 shows.

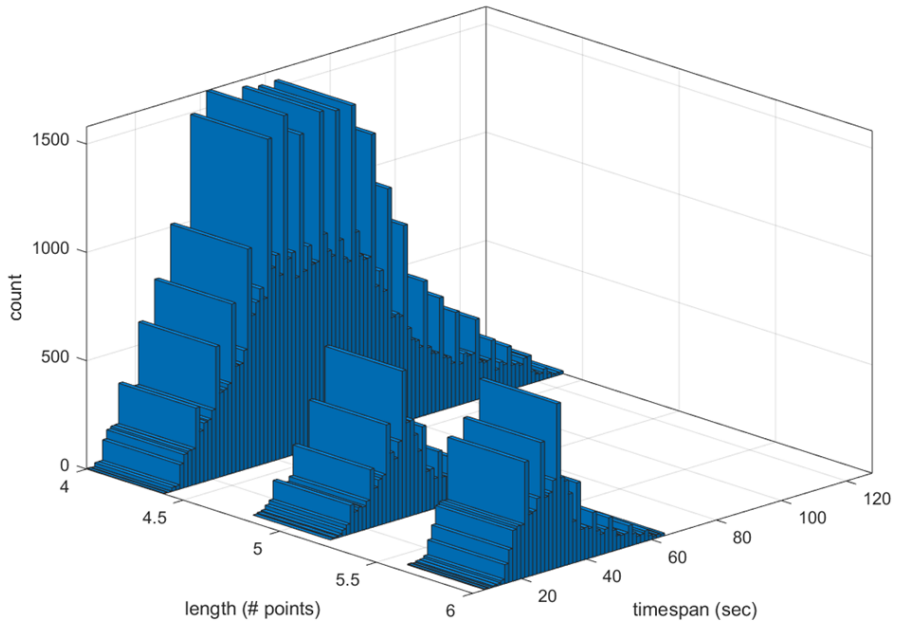

Figure 49: DTRB: Histograms of extracted slices versus data points and temporal span used.

For COM and RLM calculation, their optimal sizes were determined experimentally at $Np = 5$ and $Nr = 10$ in accordance to the DTRB configuration, i.e., the temporal size of each output 'slice'. Additionally, the value ranges for U_t, A_t and R_t were scaled down by a factor of 0.5, in order to make COM and RLM more compact and decrease the counts of such updates in their marginal bins.

The results from the reference baseline of the most comparable state-of-the-art DBP approach [191], using K-Means with Euclidean distance and speed & acceleration statistics as input, is presented in Figure 50. This is directly comparable to the TSL1 model, proposed in this study, with its results presented in Figure 51. It is clear that in the second case the clusters are significantly enhanced in terms of shape and separation, while retaining similar discrimination ratios in the dataset in terms of cluster sizes.

In the second stage of clustering, results from TSL2 are presented presented in Figure 52. Again, it is evident that with the addition of one more optimally-selected feature to the TSL1 output, the clusters become even more well-shaped and separated, almost orthogonally with the centroids essentially at the 8 corners of the 3-D hypercube.

Lastly, in the third stage of clustering, results from TSL3 in Figure 53 illustrate the useful-

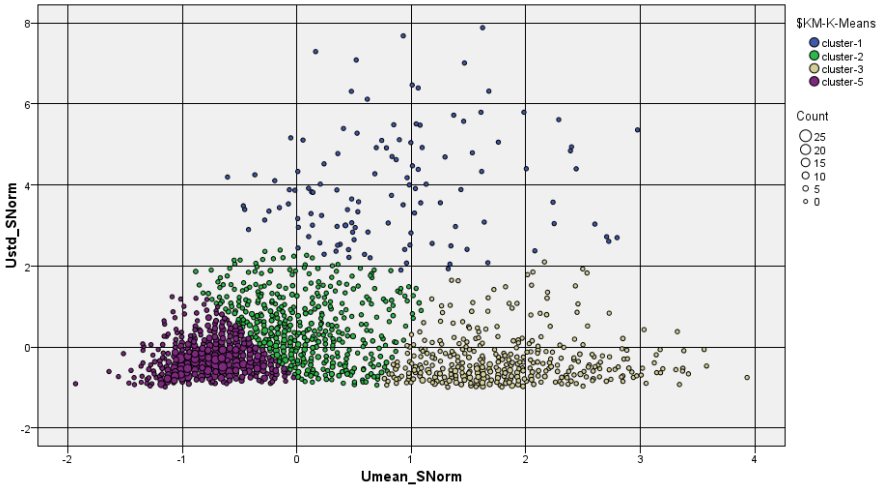

Figure 50: K-Means reference model: 4 clusters, smallest 4.2%, silhouette=0.6.

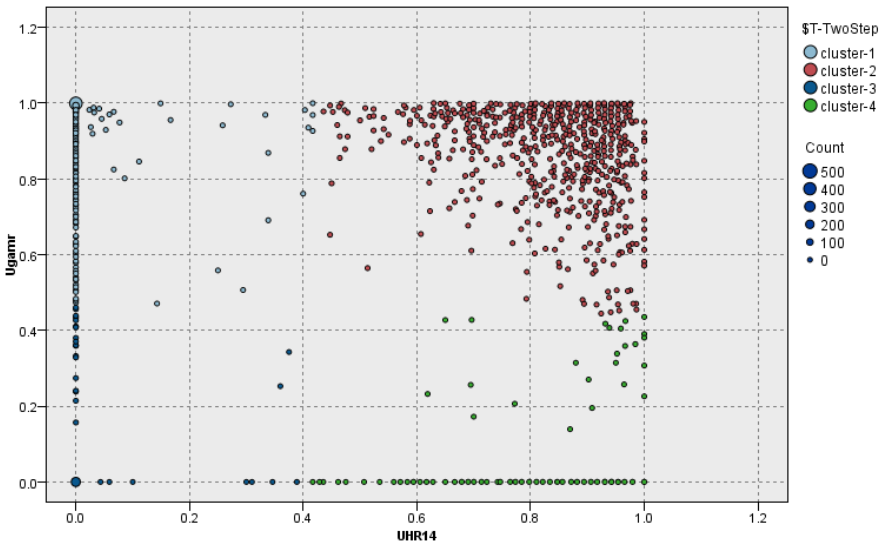

Figure 51: TSL1 model: 4 clusters, smallest 8.1%, silhouette=0.9.

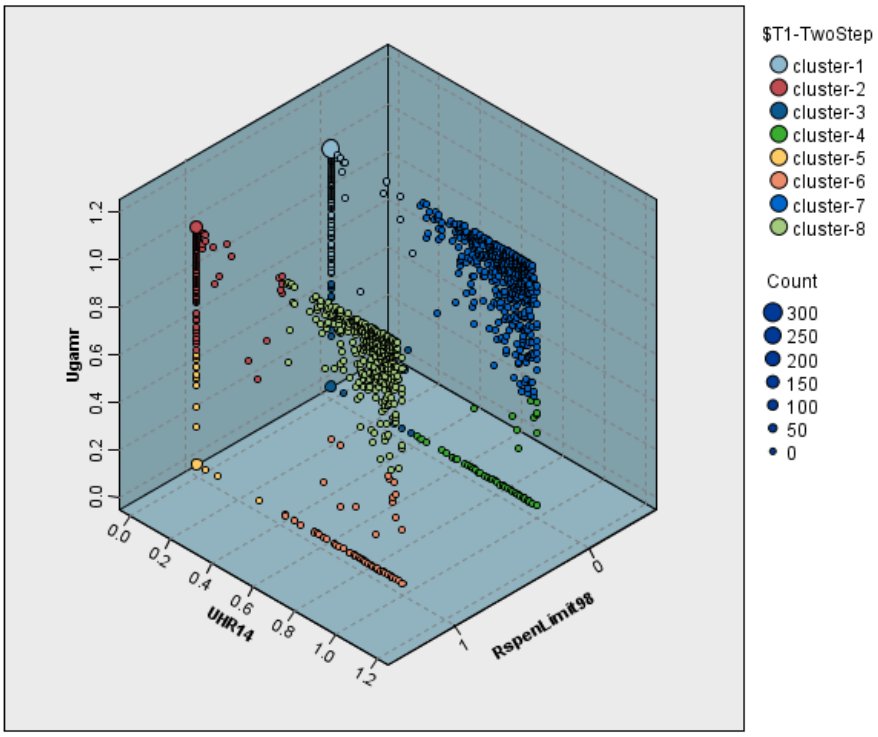

Figure 52: TSL2 model: 8 clusters, smallest 3.4%, silhouette=1.0.

ness of adding the PCA-transformed (top 2 factors used) supplementary subset of eight more optimally-selected features. Although the quality metric (silhouette) seems worse than in TSL2 and TSL1, in fact this clustering space embodies the intrinsic information content of the best 3+8 features, ranging from simple statistics to spectral model coefficients and for all three data series (speed, acceleration, turn rate), while at the same time producing well-defined clusters in a low-dimensionality space (3-D).

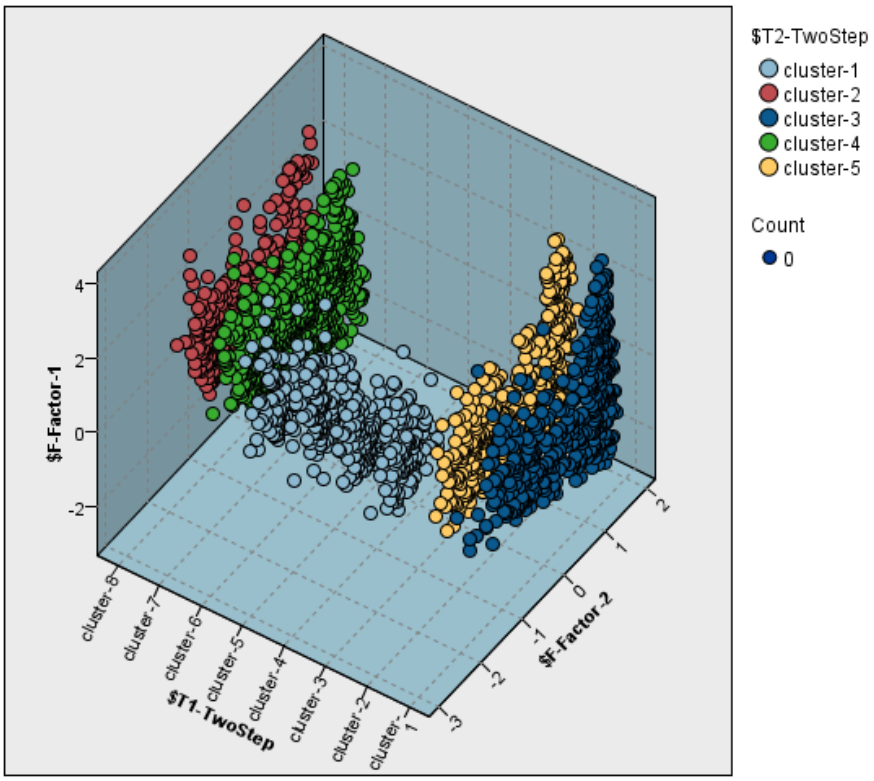

Figure 53: TSL3 model: 5 clusters (balanced), silhouette=0.3.

By examining the distribution of these features in the four TSL1 clusters it can be established that for $\{U_t^{UHR14}, U_t^{gamr}\}$: {low,low}='C3' (8.1%), {low,high}='C1' (36%), {high,low}='C4' (8.6%), {high,high}='C2' (47.3%). Furthermore, examining the mean speed distribution in the two smaller clusters which embed U_t^{gamr}='low' reveals that C3 corresponds to speeds between 0-18 km/h and C4 corresponds to speeds between 7-43 km/h. Since U_t^{UHR14} is based on (second) eigenvalue λ_2, 'high' values capture rank > 1 for COM, i.e., non-constant speed patterns. In addition, the third feature of U_t^{vpen} introduces another speed-related property of the current temporal 'slice' that is employed in TSL2.

3.5.9 Discussion

Based on the experimental results, the proposed five-phase methodology manages to successfully address all the challenges and data limitations of this DBP problem specification. DTRB together with efficient map-matching & filtering enables the necessary quality enhancement of the low-quality raw input, which otherwise would be unusable for developing the subsequent phases.

The extensive initial pool of features that are relevant to DBP patterns is dataset-agnostic and can be applied to any trajectory/location data of low temporal resolution, in order to reconstruct high-quality data series for speed, acceleration and turn rate of the vehicle. The feature selection process is also dataset-agnostic, incorporating a wide range of statistical an heuristic methods for the initial steps, then more computationally-intensive methods for identifying a subset of (combined) top-ranked features, finally applied to extensive model design via BIRCH clustering. The final result is a very small set of 3+8 features, experimentally verified as very efficient and robust DBP pattern 'encoders'.

It is important to note that the multi-level clustering approach enables the construction of models of gradual complexity in terms of computational demands for both the feature extraction and the clustering itself. Even with the three base feature functions described in Eq.8 through Eq.12 used by TSL1 and TSL2, i.e., no requirement for PCA as in TSL3, the clustering results are very clear and robust.

In order to understand what each cluster means in terms of 'good' or 'bad' driving, the intrinsic properties of the aforementioned features can be explored. In Eq.8, eigenvalue λ_2 captures the second largest 'variance' in terms of spectral component magnitude for the COM of each 'slice' of (upsampled) speed data. Strictly constant speed produces a zero-valued COM with only one non-zero cell, hence the rank of COM is 1 and $\lambda_2 = 0$. Similarly, constant acceleration produces diagonal patterns in COM with linear length proportional to the time applied, hence the rank of COM is >1 and $\lambda_2 > 0$. In other words, U_t^{HR14} captures the *existence of non-constant speed patterns* in each temporal 'slice'. In Eq.10, it is obvious that near-zero or constant speed produces $U_t^{gamr} \approx 0$ or $U_t^{gamr} \approx 1$, respectively, while non-constant speed patterns produce values in between.

Although both related to speed change, U_t^{UHR14} and U_t^{gamr} do not capture the same information content, as it is evident from TSL1 results in Figure 51. In practice the clusters can be explained in sequence as: (C1) medium/high 'stable' speed, (C2) medium/high 'volatile' speed, (C3) stopped or low speed, (C4) low/medium 'volatile' speed. From these, C3 is clearly the 'most predictable' and C2 is the 'least predictable', in relation to the criteria defined for the DBP task in Section 3.5.2. Furthermore, with the addition of U_t^{vpen} in TSL2, the 'safety' aspect is also addressed by adding another input dimension. If deemed appropriate, some of these clusters can be merged for a more coarse description, e.g. C3 and C4 (16.7%) in TSL1, thus producing 6 instead of 8 clusters in TSL2.

3.5.10 Enhancements & Future work

In this work, the DBP problem is addressed in the short-term context and with the most data-restrictive setup, using as input only low-resolution GPS location data of variable sampling rate, without any other modality available (e.g. accelerometer data). The proposed approach introduces online HMM-based map-matching to the underlying road network and robust noise filtering, as well as an algorithm for dynamic temporal resampling, to generate upsampled fixed-rate data series for speed, acceleration and turn rate.

Starting from an extensive set of feature functions, ranging from simple statistics to spectral and 'texture' analytics, the most content-rich in terms of DBP are selected. For fully unsupervised predictive modelling, a multi-stage clustering is designed and tested with a real-world dataset. The results prove the feasibility and effectiveness of the proposed approach.

Further enhancements of the proposed approach are planned in relation to the optional integration of data modalities, to exploit sensor-based acceleration instead of GPS-induced, improved clustering models, designed specifically for on-the-fly processing, as well as state-sequencing of the DBP predictive process, to enable stateful instead of stateless DBP characterization.

3.6 Hot Spot Analysis

In summary:

- *Generic question addressed*: Discover spatio-temporal cells indicated as hot spots.
- *Track&Know specific question*: Identify hot spots based on the mobility of vehicles, in order to be usedul for route optimization.
- *Novelty / Advantage over existing methods*: Previous methods mainly focus on spatio-temporal point data, not trajectories (sequences of points), and also our parallel algorithms are designed and implemented in Apache Spark so they scale gracefully for Big Data.
- *Experiments conducted*: Qualitative test on 1GB of Pilot 3 data, and scalability test on a larger collection of approximately 90GB of mobility data.
- *Type of analytics*: Descriptive analytics
- *Automation / TRL*: TRL level 3 (proof-of-concept implemented and tested, so we are between TRL 3 and 4)
- *Extension to other domains*: The method is applicable for other types of mobility data, e.g., vessels.

Huge amounts of mobility data is being produced at unprecedented rates everyday, due to the proliferation of GPS technology, the widespread adoption of smartphones, social networking, as well as the ubiquitous nature of monitoring systems. This data wealth contributes to the ever-increasing size of what is recently known as big spatial (or spatio-temporal) data [37], a specialized category of big data focusing on mobile objects, where the spatial and temporal dimensions have elevated importance. Such data include mobile objects' trajectories, but also geotagged tweets by mobile Twitter users, check-ins in Foursquare, etc. Analyzing spatio-temporal data has the potential to discover hidden patterns or result in non-trivial insights, especially when its immense volume is considered. To this end, specialized parallel data processing frameworks [3, 4, 69, 174] and algorithms [36, 192, 194, 45] have been recently developed aiming at spatial and spatio-temporal data management at scale.

In this context, a useful data analysis task is *hot spot analysis*, which is the process of identifying *statistically significant* clusters (i.e., clusters which have low probability values, based on a specific trajectory attribute). Motivated by the need for big data analytics over trajectories of vessels, we focus on discovering hot spots in the maritime domain, as this relates to various challenging use-case scenarios [32]. More specifically, having a predefined tessellation of a region into areas of interest for which there is a priori knowledge about occurring activities in them, it is very useful to be able to analyze – for instance – the intensity of the fishing activity (i.e., fishing pressure) of the areas, or to quantify the environmental fingerprint by the passage of a particular type of vessels from the areas. Similar cases exist in all mobility domains. In the aviation domain, the predicted presence of a number of aircraft above a certain threshold results in regulations in air traffic, while in the urban domain such a presence accompanied with low speed patterns implies traffic congestion. Thus, the effective discovery of such diverse types of hot spots is of critical importance for our ability to comprehend the various domains of mobility.

Our approach for hot spot discovery and analysis is based on spatio-temporal partitioning of the 3D data space in cells. Accordingly, we try to identify cells that constitute hot spots, i.e., not only do they have high density, but also that the density values are statistically significant. We employ the Getis-Ord statistic [131], a popular metric for hot spot analysis, which produces z-scores and p-values by aggregating the density values of neighboring cells. A cell is considered as a hot spot, if it is associated with high z-score and low p-value.

Unfortunately, the Getis-Ord statistic is typically applicable in the case of 2D spatial data, and even though it can be extended to the 3D case, it has been designed for point data. In contrast, our application scenario concerns trajectories of moving objects, temporally sorted sequences of spatio-temporal positions, and the applicability of hot spot analysis based on a metric, such as the Getis-Ord statistic (but also any other metric), is far from straightforward.

To this end, we formulate the problem of *Trajectory hot spot analysis* [126], where our main intuition is that the contribution of a moving object to a cell's density is proportional to the time spent by the moving object in the cell. In particular, we adapt the Getis-Ord statistic in order to capture this intuition for the case of trajectory data. Then, we propose a parallel and scalable

processing algorithm for computing hot spots in terms of spatio-temporal cells produced by grid-based partitioning of the data space under study. Our algorithm achieves scalability by parallel processing of z-scores for the different cells, and returns the exact result set. Moreover, we couple our exact algorithm with a simple approximate algorithm that only considers neighboring cells at distance h (in number of cells), instead of all cells, thus achieving significant performance improvements. More importantly, we show how to quantify the error in z-score computation, thereby developing a method that can trade-off accuracy for performance in a controlled manner.

In summary, our work makes the following contributions:

- We formulate the problem of *trajectory hot spot analysis*, by means of the popular Getis-Ord statistic, appropriately tailored to become meaningful for sequences of spatio-temporal positions, rather than plain points.
- We present a parallel algorithm that provides an exact solution to the problem, and returns spatio-temporal cells with high scores that essentially constitute hot spots.
- To improve the efficiency, we also propose an approximate parallel algorithm and a method that is able to trade-off result accuracy for computational cost, that bounds the error of the approximation in a controlled way.
- We developed our algorithms in Apache Spark and we demonstrate their efficiency and scalability by experimental evaluation on a large data set of trajectories that span three years in total.

3.6.1 Related Work

Hot spot analysis is the process of identifying statistically significant clusters. This kind of analysis is often confused with clustering with respect to the density of the identified groups [200]. The computation of density gives us information where clusters in our data exist, but not whether the clusters are statistically significant; that is the outcome of hot spot analysis. In geospatial analysis the hot spot discovery is performed with spatial statistics like Moran's I [118] or Getis-ord Gi* [131] that measure spatial autocorrelation based on feature locations and attribute values, while they result in z-scores and p-values for the predetermined points or regions of interest. A high z-score and small p-value indicates a significant hot spot. A low negative z-score and small p-value indicates a significant cold spot. The higher (or lower) the z-score, the more intense the clustering. A z-score near zero means no spatial clustering [131].

Trajectory hot spot analysis is related to the problem of finding interesting places. In [62], interesting places are defined as either: (a) the areas where a single moving object spends a large amount of time, or (b) the frequently visited areas from various trajectories, or (c) the areas where many trajectories alter their state or behavior. In [182] interesting places are identified as areas where several moving objects spend large amount of time, by moving with low speed. The minimum amount of moving objects and their minimum duration of stay, should be provided by

the user at query time. To enable efficient execution for various parameters, an indexing structure is proposed which enables fast retrieval of trajectory segments based on their speeds. Notice that these variations aim to discover spatial regions of interest, while our approach identifies interesting spatial regions for various temporal segments.

Hot spot analysis is a special case of spatio-temporal data analysis, mobility data mining and more specifically trajectory data mining, since we are interested in trajectory-based hot spot analysis. These domains have been the subject of many research efforts lately. Recent works on hot spot analysis for spatio-temporal data include [76, 109]. The study in [109] proposes a different way to visualize and analyze spatio-temporal data. The aim is to identify areas of high event density, by using multivariate kernel density estimation. Different kernels in spatial and temporal domains can be used. After such hot spots have been identified, a spatio-temporal graph can be formed to represent topological relations between hot spots. In [76], a spatio-temporal graph is analyzed in order to find anomalies on people's regular travel patterns. These interesting phenomena are called black holes and volcanos, represented as subgraphs with overall inflow traffic greater than the overall outflow by a threshold and vice-versa. The detection of frequent patterns and relations between black holes and volcanos lead to the discovery of human mobility patterns. In [92], hot spot analysis is used for studying mobile traffic. The aim is to identify locations where the density of data volumes transmitted is high, based on specific values of thresholds. The results of the analysis are then used to detect the distribution of mobile data traffic hot spots and to propose a meaningful cell deployment strategy.

In the domain of trajectory data mining [202] there are several clustering approaches that are relevant to this work. The typical approach is to either transform trajectories to vector data, in order for well-known clustering algorithms to be applicable, or to define appropriate trajectory similarity functions, which is the basic building block of every clustering approach. For instance, CenTR-I-FCM [141] builds upon a Fuzzy C-Means variant to perform a kind of time-focused local clustering using a region growing technique under similarity and density constraints. For each time period, the algorithm determines an initial seed region (that corresponds to the sub-trajectory restricted inside the period) and searches for the maximum region that is composed of all sub-trajectories that are similar with respect to a distance threshold d and dense with respect to a density threshold δ. Subsequently, the growing process begins and the algorithm tries to find the next region to extend among the most similar sub-trajectories. The algorithm continues until no more growing can be applied, appending in each repetition the temporally local centroid. In the same line of research, having defined an effective similarity metric, TOPTICS [123] adapts OPTICS [10] to enable whole-trajectory clustering (i.e., clustering the entire trajectories), TRACLUS [98] exploits on DBSCAN [40] to support sub-trajectory clustering, while T-Sampling [142, 134], introduces trajectory segmentation (aiming at temporal-constrained sub-trajectory clustering [143]), by taking into account the neighborhood of a trajectory in the rest of the data set, yielding a segmentation that is related only on the number of neighboring segments that vote for the line segments of a trajectory as the most representatives. All the

above trajectory clustering approaches they are capable of identifying trajectory clusters and their densities but do not tackle the issue of statistical significance in the space-time they take place.

There are several other methods that try to identify frequent (thus dense) trajectory patterns. In case where moving objects move under the restrictions of a transportation network, [159] proposed an online approach to discover and maintain hot motions paths while [28] tackled the problem of discovering the most popular route between two locations based on the historical behavior of travelers. In case where objects move without constraints, [25] proposed a method to discover collocation patterns, while in [56] where the goal is to discover sequential trajectory patterns (T-patterns), the popular regions that participate in T-patterns can be computed automatically following a clustering approach that utilizes the density of the trajectories in the space. The algorithm starts by tessellating the space into small rectangular cells, for each of which the density (i.e., the number of trajectories that either cross it or found inside the cell) is computed. Then, by following a region-growing technique the dense areas are enlarged by including nearby cells as long as the density criterion is fulfilled. The problem of identifying hot spots from trajectory data in indoor spaces has been studied in [85]. It introduces a formula for computing scores for indoor locations based on users' interests rather than measuring the amount of time a user spends in a specific area. This formula is used for calculating a score for each indoor area, based on the mutual reinforcement relationship between users and indoor locations. These methods have a different focus from our proposal sharing similar objectives and shortcomings as the aforementioned techniques.

The problem of finding hot spots for spatio-temporal point data has been studied in SigSpatial Cup at 2016 (http://sigspatial2016.sigspatial.org/giscup2016/home) where several interesting methods where proposed. Among others, [113, 125] proposed algorithms to identify hot spots based on spatial density of point data (in particular drop-off locations of taxis). Instead, this work, we study the problem of hot spot analysis for trajectory data, which is different because the effect of a data point to a cell depends on the trajectory in which it belongs to (i.e., on other points). To the best of our knowledge, there is a lack of parallel processing solutions that operate on distributed trajectory data in an efficient and scalable way to discover trajectory-based hot spots.

3.6.2 Problem Formulation

Consider a moving object database $\mathcal{D}$ that consists of trajectories $\tau \in \mathcal{D}$ of moving objects. A trajectory, denoted by an identifier τ, is a sequence of data points p described by 2D spatial coordinates ($p.x$ and $p.y$), a timestamp ($p.t$), as well as any other information related to the spatio-temporal position of p. For example, attributes referring to weather information. We also use $p.\tau$ to refer to the trajectory that p belongs to. Furthermore, consider a spatio-temporal partitioning $\mathcal{P}$ which partitions the 3D spatio-temporal domain into n 3D cells $\{c_1, \ldots, c_n\} \in \mathcal{P}$. Each data point p is mapped to one cell c_i, which is determined based on p being enclosed in c_i.

Symbol	Description
$\mathcal{D}$	Spatio-temporal data set
$p \in \mathcal{D}$	Spatio-temporal data point $(p.x, p.y, p.t)$
τ	A trajectory (as a sequence) of points $p_1, p_2, \ldots$
$\mathcal{P}$	3D space partitioning $\mathcal{P} = \{c_1, \ldots, c_n\}$
c_i	The i-th cell of partitioning $\mathcal{P}$, $(1 \leq i \leq n)$
x_i	Attribute value of cell $c_i \in \mathcal{P}$
$w_{i,j}$	A weight indicating the influence of cell c_j to c_i
α	Parameter used to define the weights $w_{i,j}$
n	The number of cells in $\mathcal{P}$
G_i^*	The Getis-Ord statistic for cell c_i
$\hat{G}_i^*$	An approximate value of G_i^*
$\mathcal{H}_{ij}$	Distance between cells c_i and c_j (in number of cells)
top-k	Requested number of most significant cells

Table 9: Overview of symbols.

Also, we use $c_{i_{start}}$, $c_{i_{end}}$ to refer to temporal start and end of a cell c_i.

We also define the *attribute value* x_i of the cell c_i as: $x_i = \sum_{\tau \in c_i} \frac{t_{end} - t_{start}}{c_{i_{end}} - c_{i_{start}}}$, thus each trajectory τ that exists in a spatio-temporal cell c_i contributes to the cell's attribute value by its temporal duration $t_{end} - t_{start}$ in c_i normalized by dividing with the cell's temporal lifespan $c_{i_{end}} - c_{i_{start}}$. This definition implies that the longer a moving object's trajectory stays in a spatio-temporal cell, the higher its contribution to the cell's hot spot value.

The problem of *hot spot analysis* addressed in this work is to identify statistically significant spatio-temporal areas (i.e., cells of $\mathcal{P}$), where the significance of a cell c_i is a function of the cell's attribute value x_i, but also of other neighboring cells' attribute values. A commonly used function (statistic) is the Getis-Ord statistic G_i^*, defined as [131]:

$$G_i^* = \frac{\sum_{j=1}^{n} w_{i,j} x_j - \overline{X} \sum_{j=1}^{n} w_{i,j}}{S \sqrt{\frac{[n \sum_{j=1}^{n} w_{i,j}^2 - (\sum_{j=1}^{n} w_{i,j})^2]}{n-1}}} \tag{19}$$

where x_j is the attribute value for cell j, $w_{i,j}$ is the spatial weight between cell i and j, n is equal to the total number of cells, and:

$$\overline{X} = \frac{\sum_{j=1}^{n} x_j}{n} \tag{20}$$

$$S = \sqrt{\frac{\sum_{j=1}^{n} x_j^2}{n} - (\overline{X})^2} \tag{21}$$

To model the intuition that the influence of a neighboring cell c_i to a given cell c_j should be decreasing with increased distance, we employ a weight function that decreases with increasing distance. Namely, we define: $w_{ij} = a^{1-\rho}$, where $a > 1$ is an application-dependent parameter,

and ρ represents the distance between cell i and cell j measured in number of cells[6]. For immediate neighboring cells, where $\rho = 1$, we have $w_{ij} = 1$, while for the next neighbors we have respectively: $1/a$, $1/a^2$, ...,. This definition captures an "exponential decay" with distance in the contribution of neighboring cells to a given cell.

Based on this, the problem of trajectory hot spot analysis is to identify the k most statistically significant cells according to the Getis-Ord statistic, and can be formally stated as follows.

Problem 6 *(Trajectory hot spot analysis) Given a trajectory data set $\mathcal{D}$ and a space partitioning $\mathcal{P}$, find the top-k cells $TOPK = \{c_1, \ldots, c_k\} \in \mathcal{P}$ based on the Getis-Ord statistic G_i^*, such that: $G_i^* \geq G_j^*$, $\forall c_i \in TOPK$, $c_j \in \mathcal{P} - TOPK$.*

In this work, we study an instance of Problem 6, where the aim is to perform hot spot analysis for trajectories over massive spatio-temporal data by proposing a parallel and scalable solution. Thus, we turn our attention to large-scale trajectory data sets that exceed the storage and processing capabilities of a single centralized node. We assume that the data set $\mathcal{D}$ is stored distributed in multiple nodes, without any more specific assumptions about the exact partitioning mechanism. Put differently, a node stores a subset D_i of the records of $\mathcal{D}$, and it holds that $D_i \bigcap D_j = \emptyset$ (for $i \neq j$), and $\bigcup D_i = \mathcal{D}$. Hence, our work studies a distributed version of Problem 6. Table 9 provides an overview of the notation used for the work on hot spot analysis.

3.6.3 An Exact Algorithm: THS

In this section, we present the THS (Trajectory Hot Spot) algorithm for distributed hot spot analysis over big trajectory data. The proposed algorithm is designed to be efficiently executed over a set of nodes in parallel and is implemented in Apache Spark. The input data set $\mathcal{D}$ is assumed to be stored in a distributed file system, in particular HDFS.

3.6.3.1 Overview Intuitively, our solution consists of three main steps, which are depicted in Figure 54. In the first step, the goal is to compute all the cells' attribute values of a user defined spatio-temporal equi-width grid. To this end, the individual attribute values of trajectory data points are aggregated into cell attribute values, using the formula introduced in the previous subsection. Then, during the second step, we calculate the cells' attribute mean value and standard deviation which will be provided to the Getis-Ord formula later. Furthermore, we compute the weighted sum of the values for each cell c_i: $\sum_{j=1}^{n} w_{i,j}x_j$, by exchanging the cells' attribute values between themselves. Upon successful completion of the second step, we have calculated all the individual variables included in the Getis-Ord formula, and we are now ready to commence the final step. The goal of the third step is to calculate the z-scores of the spatio-temporal grid cells, by applying the Getis-Ord formula. The trajectory hot spots can then

[6]Notice that this distance applies to the 3D grid, i.e., cells with the same spatial extent that belong to different temporal intervals have non-zero distance.

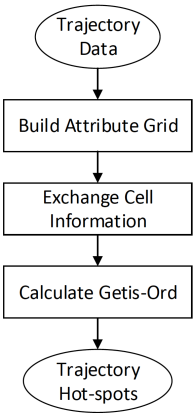

Figure 54: Overview of THS algorithm.

be trivially calculated, by either selecting the top-k cells with the higher z-score values, or by selecting the cells having a p-value below a specified threshold.

The above description explains the rationale of our approach. In the following, we describe the implementation of our solution using Apache Spark Core API, along with the necessary technical details.

Algorithm 6 THS Step 1: Build Attribute Grid

1: **Input:** $\mathcal{D}$, $\mathcal{P}$
2: **Output:** gridRDD: RDD[i, x_i]
3: **function:**
4: gridRDD = $\mathcal{D}$.mapToPair(p =>
5: emit **new** pair(getCellId(p) $\oplus$ $p.\tau$, $p.v$)
6:).reduceByKey(t_1, t_2 =>
7: emit **new** pair(MIN(t_1, t_2), MAX(t_1, t_2))
8:).mapToPair(cell_trajectory_id, (t_{start} , t_{end}) =>
9: emit **new** pair(cell_id, $\frac{t_{end}-t_{start}}{c_{i_{end}}-c_{i_{start}}}$)
10:).reduceByKey(v_1, v_2 => emit $v_1 + v_2$)
11: **end function**

3.6.3.2 Building the Attribute Grid The first step of THS, depicted in Algorithm 6, takes as input a data set $\mathcal{D}$ of trajectories stored in HDFS and a spatio-temporal partitioning $\mathcal{P}$, which defines the size of all cells c_i regarding their spatial and temporal dimensions. We use a function getCellId(p) to get the identifier i of cell c_i enclosing data point p. Initially, the trajectory data points are mapped to key value pairs (line 5), where the key is composed by a string concatenation (denoted with $\oplus$ in the algorithm) of the data point's cell id and its trajectory id ($p.\tau$), while the value is the timestamp of p. This assignment of composite keys, enables us to group data points by cell id and trajectory id; we calculate the minimum and maximum attribute values (t_{start}, t_{end} respectively) for each such group (line 7). Then, we

compute the fraction $\frac{t_{end}-t_{start}}{c_{i_{end}}-c_{i_{start}}}$, individually for each group defined by the composite keys, and keep only the cell id part of the key (line 9). We perform a regrouping based on the new keys, to calculate the sum of the fractions for each cell (line 10). Hence, we have now successfully built the attribute grid (*gridRDD*), by computing each cell's attribute value $x_i = \sum_{t \in c_i} \frac{t_{end}-t_{start}}{c_{i_{end}}-c_{i_{start}}}$.

Algorithm 7 THS Step 2: Exchange Cell Information

1: **Input:** gridRDD: RDD[i, x_i], $\mathcal{P}$
2: **Output:** $\overline{X}$, S, wSumRDD: RDD[i, $\sum_{j=1}^{n} w_{i,j}x_j$]
3: **function:**
4: gridRDD.forEach(x_i => update accumulators)
5: calculate $\overline{X}$ and S from accumulators
6: wSumRDD = gridRDD.flatMapToPair(i, x_i =>
7: w_i = getWeightList(i)
8: **for each** j **in** w_i **do**
9: emit **new** pair(j, $x_i * w_{i,j}$)
10: **end for**
11:).reduceByKey(wx_1, wx_2 => emit $wx_1 + wx_2$)
12: **end function**

3.6.3.3 Exchanging Cell Information In its second step, as presented in Algorithm 7, THS takes as input the attribute grid produced by the first step and the spatio-temporal partitioning $\mathcal{P}$. In line 4, we distributively calculate the sum and squared sum of the cells' attribute values. Having computed these sums, we can trivially calculate the mean value $\overline{X}$ and standard deviation S, in a centralized fashion (line 5).

Then, our goal is to broadcast each cell's weighted attribute value to all the other cells of the grid. To this end, in lines 7-10, we first get the list of weights between current cell c_i and all the cells c_j of the grid (line 7). For each cell c_j we emit a new key value pair consisting of the j value as the key and the weighted attribute as its value (line 9). Then, we perform a grouping of these key value pairs, based on their keys (i.e., cell ids), while calculating the sum of their weighted attribute values (line 11). The result of this operation is the weighted sum grid (*wSumRDD*), which will be used for computing the Getis-Ord formula.

Algorithm 8 THS Step 3: Calculate Getis-Ord

1: **Input:** $\overline{X}$, S, wSumRDD: RDD[i, $\sum_{j=1}^{n} w_{i,j}x_j$], $\mathcal{P}$
2: **Output:** G_iRDD: RDD[i, G_i^*]
3: **function:**
4: G_iRDD = wSumRDD.mapToPair(i, wx_i =>
5: sum_i = getWeightSum(i)
6: $squaredSum_i$ = getWeightSquaredSum(i)
7: emit **new** pair(i, $\frac{wx_i - \overline{X} \cdot sum_i}{S\sqrt{\frac{[n \cdot squaredSum_i - (sum_i)^2]}{n-1}}}$)
8:)
9: **end function**

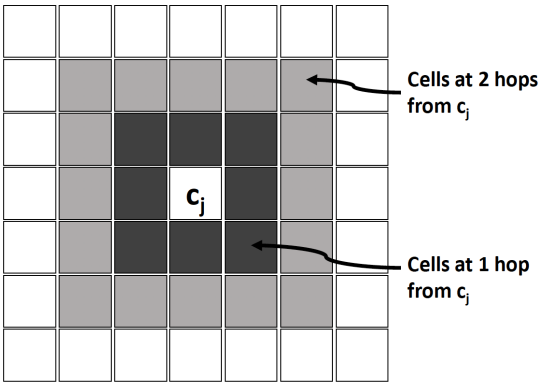

Figure 55: Example of cells at distance from a reference cell c_j (the dark color indicates the weight of their contribution to c_j's value x_j).

3.6.3.4 Calculating z-scores with Getis-Ord statistic The third step of THS is depicted in Algorithm 8. It takes as input the $\overline{X}$ and S values computed in the previous part, along with the weighted sum grid and the spatio-temporal partitioning $\mathcal{P}$. We map each cell's weighted sum attribute value to a z-score by applying the Getis-Ord formula. The sum and squared sum of weights are initially computed (lines 5,6) in order to be provided to the calculation of the Getis-Ord formula right after (line 7). Finally, the result of this operation, results to a data set ($G_i RDD$) consisting of cell ids and their Getis-Ord z-scores.

3.6.4 An Approximate Algorithm: aTHS

The afore-described algorithm (THS) is exact and computes the correct hot spots over widely distributed data. However, its computational cost is relatively high and can be intolerable when the number of cells n in $\mathcal{P}$ is large. This is because every cell's value must be sent to all other cells of the grid, thus leading to data exchange through the network of $O(n^2)$ as well as analogous computational cost, which is prohibitive for large values of n.

Instead, in this section, we propose an approximate algorithm, denoted aTHS, for solving the problem. The rationale behind aTHS algorithm is to compute an approximation $\hat{G}_i^*$ of the value G_i^* of a cell c_i, by taking into account only those cells at maximum distance h from c_i. The distance is measured in number of cells. Intuitively, cells that are located far away from c_i will only have a small effect on the value G_i^*, and should not affect its accuracy significantly when neglected.

More interestingly, we show how to quantify the error $\Delta G_i^* = G_i^* - \hat{G}_i^*$ of the computed hot spot z-score of any cell c_i, when taking into account only neighboring cells at distance h. In turn, this yields an analytical method that can be used to trade-off accuracy with computational efficiency, having bounding error values.

3.6.4.1 The aTHS Algorithm Based on the problem definition, cells located far away from a reference cell c_i, only have a limited contribution to the Getis-Ord value G_i^* of c_i. Our approximate algorithm (aTHS) exploits this concept, and can be parameterized with a value h, which defines the subset of neighboring cells that contribute to the value of c_i. We express h in terms of cells, for instance setting h=2 corresponds to the case depicted in Figure 55, where only the colored cells will be taken into account by aTHS for the computation of $\hat{G}_i^*$ (an approximation of the value of G_i^*). In practice, the relationship between cell c_j and white cells can be expressed by setting their weight factor equal to zero.

Algorithm 9 aTHS Part 2: Exchange Cell Information

1: **Input:** gridRDD: RDD[i, x_i], $\mathcal{P}$, h
2: **Output:** $\overline{X}$, S, wSumRDD: RDD[i, $\sum_{j=1}^{n} w_{i,j}x_j$]
3: **function:**
4: gridRDD.forEach(x_i => update accumulators)
5: calculate $\overline{X}$ and S from accumulators
6: wSumRDD = gridRDD.flatMapToPair(i, x_i =>
7: w_i = getWeightList(i)
8: **for each** j **in** w_i **do**
9: **if** DISTANCE(i, j) $\leq h$ **then**
10: emit **new** pair(j, $x_i * w_{i,j}$)
11: **end if**
12: **end for**
13:).reduceByKey(wx_1, wx_2 => emit $wx_1 + wx_2$)
14: **end function**

In algorithmic terms, aTHS is differentiated from THS in the second and third step. Algorithm 9 describes the pseudo-code of the second step of aTHS. The main difference is in line 9, where we check the distance between cells i and j; if the distance is greater than the threshold h, then the emission of a new key value pair does not occur (i.e., we do not broadcast the weighted attribute value of cell c_i to cell c_j).

The third step of aTHS calculates the weight attribute sum and squared sum, by applying a weight factor equal to zero, for cells c_j which are located in a distance farther than h from c_i. This change affects only the implementation of the *getWeightSum* and *getWeightSquaredSum* methods, used in lines 5,6 of Algorithm 8.

3.6.4.2 Controlling the Error ΔG_i^* In this section, we present an upper bound for the value ΔG_i^* (we refer the reader to [126] for the derivation of this result):

$$\Delta G_i^* \leq \frac{x_{max} - \overline{X}}{\gamma} \sum_{\rho > h} a^{1-\rho}[f(\rho, d) - f(\rho - 1, d)]$$

In summary, we can compute an upper bound for the error ΔG_i^* introduced to the Getis-Ord value of a cell c_i, due to approximate processing using only neighbors at distance h. In turn, this

allows us to explicitly set the value h in Algorithm aTHS, in such a way that we can guarantee that the maximum error introduced is quantified. In practice, an analyst can exploit our method to trade-off accuracy for computational efficiency, making aTHS an attractive algorithm for trajectory hot spot analysis over massive data sets.

3.6.5 Empirical Evaluation

In this section, we report the results of our empirical study on data of Pilot 3 of the project. The data set used in the experiment consists of the positions reported by 56 vehicles for a period of 1 year in the wider area of Greece. The size of this data set is approximately 1GB, so it is reasonably large.

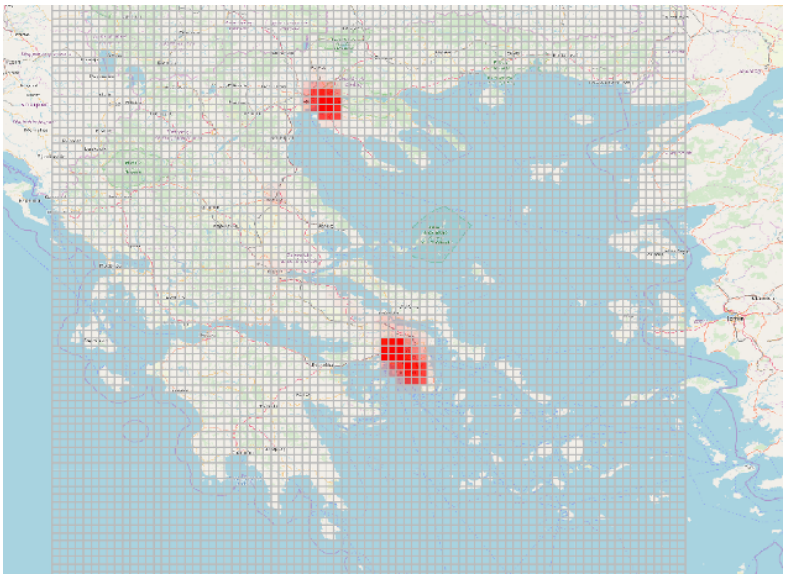

Figure 56: Hot spots discovered in the wider area of Greece: most hot spot cells are located in Athens and Thessaloniki.

Figures 56–58 depict results derived by applying our technique for hot spot discovery on the aforementioned data set. Different shades of red correspond to the hot spot value, so more intense shades of red indicate higher hot spot value. Figure 56 indicates that most hot spots are located in Athens and Thessaloniki, the two largest cities in Greece. This is reasonable, since most of the data is located in these cities, so the result is correct but of limited use.

When we focus in the wider area of Athens, depicted in Figure 57, the results are more meaningful, since one can clearly see hot spots in Koropi and Aspropirgos. Other hot spots can also be recognized in Egaleo, Palaio Faliro and Vouliagmeni (in pink). Also, one can recognize that different main avenues of the city seem to contribute to hot spots. In this chart, we depict all hot spots that were identified, not the top-k hot spots.

Figure 58 shows the top-50 hot spots without using different shades of red, so other less

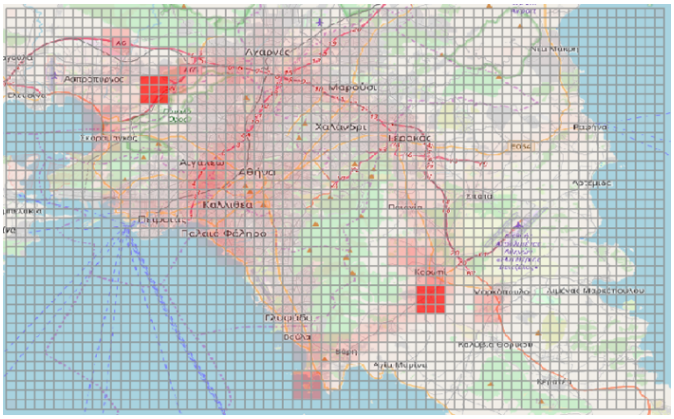

Figure 57: Hot spots discovered when focusing on the area of Athens.

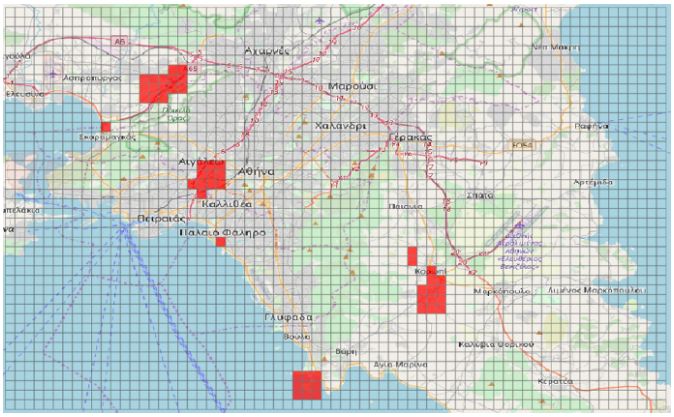

Figure 58: Top-50 hot spots discovered when focusing on the area of Athens.

significant hot spots are suppressed. This also signifies the importance of computing a ranked list of hot spots, since a data analyst can retrieve only the top-k cells to obtain more focused results.

Regarding performance results and scalability, we refer to the experiments reported in our published paper [126] in IEEE BigData'18. In summary, we report results using a data set of approximately 90GB of spatio-temporal data in a cluster using 10 physical nodes, running Apache Spark. The results show that our algorithms can compute hot spots over large data sets in a few hundreds of seconds, depending on different parameters, such as spatial cell size and temporal cell size. Also, the approximate algorithm aTHS is indeed able to speed up computation when using a smaller value of h.

3.6.6 Summary and Future Work

In this work, we formulate the problem of Trajectory hot spot analysis, which finds many real-life applications in the case of big trajectory data. We proposed two parallel data processing algorithms that solve the problem, which are scalable for data of large volumes. Our first algorithm (THS) provides an exact solution to the problem, but may be computationally expensive depending on the underlying grid partitioning and the number of cells. Also, we propose an approximate algorithm (aTHS) that practically ignores the contribution of cells located further away from the cell under study, thereby saving computational cost. Perhaps more importantly, we propose a method that can be used to bound the error in the approximation, for a given subset of cells that are taken into account. Thus, we can trade-off accuracy for efficiency in a controlled manner. Our implementation is based on Apache Spark, and we demonstrate the scalability of our approach using real-life data.

In our future work, we intend to explore variants of the problem of trajectory hot spot analysis, which is a problem that deserves further study. More concretely, we will address online hot spot detection in a streaming context, where the aim is to detect hot spots and significant changes in existing hot spots as new data arrive into the system. Another promising research direction is extending this work to road networks, aiming to identify hot spots in the form of edges or paths of the underlying graph representing the road network, instead of 3D regions in the space-time domain.

3.7 Identifying business activity-travel patterns based on GPS data

In summary:

- *Generic question addressed*: Identifying business activity-travel patterns based on GPS data.
- *Track&Know specific question*: Big mobility data pattern detection.
- *Novelty / Advantage over existing methods*: Business activity and travel features have been described on a qualitative level; no methods exist yet to quantitatively analyse business travel behaviour. The developed method here is novel in terms of identifying typical activity-travel patterns associated with each business vehicle type and quantifying (using numeric scores) how similar or different the travel practice of a specific company (or vehicle) is to the typical behaviour of the corresponding vehicle type.
- *Experiments conducted*: Testing on a TrackKnow Pilot dataset with quantitative and qualitative evaluations.
- *Type of analytics*: Descriptive analytics.
- *Automation / TRL*: TRL level 3 (proof-of-concept implemented and tested)
- *Extension to other domains*: The method is applicable for other types of mobility data, e.g. mobile phone data.

3.7.1 Problem statement

As employers, suppliers and transport providers, organizations are responsible for the generation of a large part of traffic flows on the transport network. However, despite the significance of business travel to overall mobility, the underlying activity compositions of the movement and the decision making process within organizations have not been well understood. This lack of knowledge contrasts with a large body of work undertaken over the past years, which has developed a theoretical and applied evidence base (e.g. activity-based travel demand models) regarding personal travel behavior. Personal and business travel behavior share similarities but also display differences. Both behavior exhibits a certain degree of spatial-temporal regularities and sequential ordering across days. However, the types of activities performed and the duration and visit frequency of the stops are different, while the factors based on which travel decisions are made also vary. As a result, further research needs to be done in order to go from personal activity-travel patterns up to business behavioral mobility knowledge, capable of supporting policies that are related to business travel. This is exactly the challenge, and if a method can be

found which helps to bridge this gap, the potential application of using the behavioral knowledge is considerable. Particularly, given the increasing attention on the environmental effects caused by business vehicles, both governments and organizations face problems of addressing a radically changed economic landscape, pressing environmental issues, increasing driving safety, and managing travel demand more efficiently. An improved behavioral mobility understanding will thus help uncover business activities and travel features and explore factors that would lead to addressing these problems and enhancing business travel in the aimed directions.

To this end, a new method has been proposed in this subsection, aimed to identify typical activity-travel patterns from business travel and characterize travel behavior of specific companies or vehicles (and corresponding drivers) based on the derived patterns. Business travel refers to trips made in the course of work for job-related obligations, and in this study travel on road (instead of by rail or sea) is considered. Particularly, trips made by vehicles of six commonly used types, including cars, buses, vans, trucks (including trucks-35t and trucks-3ax belonging to medium- and heavy-duty trucks respectively), and motors (motorcycles), are analyzed. Vehicles of these types and corresponding trips support various business activities and generate an array of services, playing a pivotal role in the economy [11]. Cars enable location changes of personnel to meet internal or external partners for business meetings or customer services, whereas buses transport a group of passengers (e.g. school students or tourists) between spatially distributed origins and destinations. Vans provide postal and courier services and mobile workshops, motors deliver small items (e.g. food), while trucks transport large-size and heavy goods (e.g. stocks, electronics and construction materials) or are used for essential public services (e.g. garbage collection). The activities performed by different types of vehicles vary to a certain degree, in terms of the spatial-temporal features (e.g. the visit frequency and visit time, activity duration, travel time required, and sequential order) of the activities. The differences in these features further lead to deviations in travel behavior associated with these vehicle types. In the proposed method, the spatial-temporal features of stops (obtained from GPS trajectories of business vehicles) are extracted, based on which stops are classified. Daily travel sequences composed of the classified stops are subsequently formed. Activity-travel patterns are further identified from the sequences of each vehicle type, from which travel behavior associated with the corresponding type is inferred.

3.7.2 The proposed method

Specifically, the approach consists of five major steps as follows. (1) GPS data collected from business vehicles are preprocessed and stops are extracted. (2) A set of spatial-temporal variables characterizing each stop is defined, and Feature Selection Techniques are applied to choose the most effective variables. Upon these selected variables, stops are classified. (3) The obtained stops from each vehicle each day, along with their sequential order, form a stop-sequence, and sequences of all vehicles of the same vehicle type are searched for stop-patterns by means of Sequential Pattern Mining. (4) The derived patterns are further clustered based on Sequence

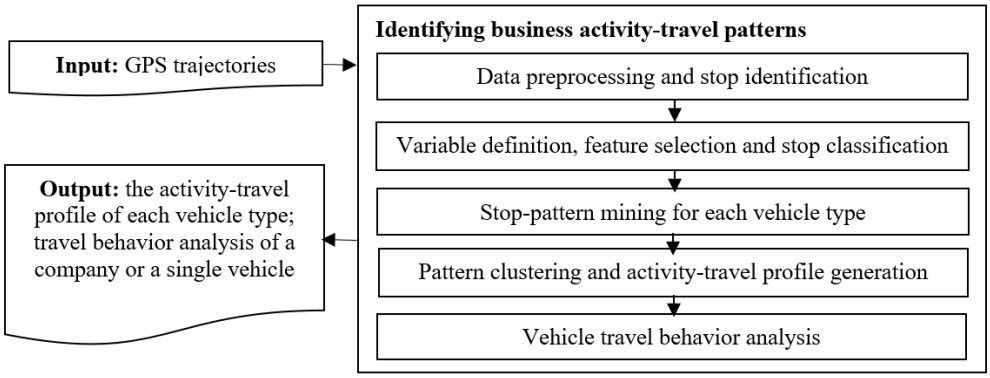

Figure 59: The overall structure of the method

Alignment Methods; the resultant clusters, in combination with the sequence support of each cluster, defines an activity-travel profile. (5) This profile characterizes typical travel behavior performed by the corresponding vehicle type, based on which specific travel features of a set of new vehicles are analyzed. Figure 59 outlines the approach, and the details can be referred to in the paper [42].

3.7.3 Experimental results

The pilot use case 3 provided by the Vodafone Group are adopted for the experiment.

(1) Stop classification

Four variables characterizing key features of each stop were picked up by the feature selection technique, including FreqPerDay (the probability to visit the stop each day), Duration (the average duration over all visits to the stop), TimeFirst (the earliest time-of-day at the stop), and TravelTime (the average travel time required between the stop and its previous stop). These variables maximum distinguish stops made by different types of vehicles in terms of the spatial-temporal dimensions, and they are further categorized using significant cutting values of these variables (See Figure 60).

Based on these variables and their discrete values, stops are classified (See Figure 61). In total, 36 stop classes are generated; an extra class '10' is defined representing work places.

(2) The obtained stop-patterns and clusters

Figure 62 presents the clusters for three vehicle types including vans, cars and trucks-3ax. In this table, each row represents a cluster, and the columns describe the properties of the cluster. They include: the vehicle type (Type); major stop class (MC) (the most frequent class in the patterns of the cluster); secondary stop classes (SC) (the remaining classes); the longest pattern ('Longest P') and this pattern's length (Len) (the number of stops in this pattern), sequence support (Su) (the number of stop-sequences containing this pattern relative to that of all sequences being searched for); the number of all patterns in the cluster (N) and the sequence

$$\text{FreqPerDay} = \begin{cases} 1\ (\text{low}),\ \textit{if}\ \text{FreqPerDay} < 0.02 \\ 2\ (\text{middle}),\ \textit{if}\ \text{FreqPerDay} \geq 0.02\ \text{and}\ \text{FreqPerDay} < 0.16 \\ 3\ (\text{high}),\ \textit{if}\ \text{FreqPerDay} \geq 0.16 \end{cases}$$

$$\text{Duration} = \begin{cases} 1\ (\text{short}),\ \textit{if}\ \text{Duration} < 25\,\text{min} \\ 2\ (\text{middle}),\ \textit{if}\ \text{Duration} \geq 25\,\text{min}\ \text{and}\ \text{Duration} < 140\,\text{min} \\ 3\ (\text{long}),\ \textit{if}\ \text{Duration} \geq 140\,\text{min} \end{cases}$$

$$\text{TimeFirst} = \begin{cases} 1\ (\text{morning}),\ \textit{if}\ \text{TimeFirst} < 10:30\textit{am} \\ 2\ (\text{day}),\ \textit{if}\ \text{TimeFirst} \geq 10:30\textit{am} \end{cases}$$

$$\text{TravelTime} = \begin{cases} 1\ (\text{short}),\ \textit{if}\ \text{TravelTime} < 60\,\text{min} \\ 2\ (\text{long}),\ \textit{if}\ \text{TravelTime} \geq 60\,\text{min} \end{cases}$$

Figure 60: Variable cutting values

Class	Freq	Dur	First	Tra	**Class**	Freq	Dur	First	Tra	**Class**	Freq	Dur	First	Tra
11	1	1	1	1	**41**	2	1	1	1	**71**	3	1	1	1
12	1	1	1	2	**42**	2	1	1	2	**72**	3	1	1	2
13	1	1	2	1	**43**	2	1	2	1	**73**	3	1	2	1
14	1	1	2	2	**44**	2	1	2	2	**74**	3	1	2	2
21	1	2	1	1	**51**	2	2	1	1	**81**	3	2	1	1
22	1	2	1	2	**52**	2	2	1	2	**82**	3	2	1	2
23	1	2	2	1	**53**	2	2	2	1	**83**	3	2	2	1
24	1	2	2	2	**54**	2	2	2	2	**84**	3	2	2	2
31	1	3	1	1	**61**	2	3	1	1	**91**	3	3	1	1
32	1	3	1	2	**62**	2	3	1	2	**92**	3	3	1	2
33	1	3	2	1	**63**	2	3	2	1	**93**	3	3	2	1
34	1	3	2	2	**64**	2	3	2	2	**94**	3	3	2	2

Note: the column 'Class' is the assigned class values; while 'Freq', 'Dur', 'First' and 'Tra' represent the variables *FreqPerday*, *Duration*, *TimeFirst* and *TravelTime*, respectively.

Figure 61: Stop classification and corresponding categorized values of the classification variables

support of the cluster (SuN) (the sum of Su over all the pattern); and the category (defined in the next subsection).

(3) Cluster analysis and differences in clusters across vehicle types

The clusters across different vehicle types describe varied activity and travel features of business trips, and they can be further grouped into four categories as follows.

Category 1 (a chain of stops): including clusters with the major class MC =('43', '53', '73', '83', '93'). This category depicts patterns consisting of a chain of multiple stops between two consecutive work locations, and these stops are characterized with a middle-high visit frequency (FreqPerDay>0.02), short travel time (TravelTime<60min) and being conducted during the day (after 10:30am).

Each of the chains represents a sequence of stops that are performed in the corresponding temporal order frequently. The chain can represent a combination of specific stop locations which

Type	MC	SC	Longest P	Len	Su	N	SuN	Category
Vans	73	43, 83	10-73-73-73-73-73-73-73-73-73-73-73-73-73-10	15	0.10	120	10.88	1
	83	53, 73	10-83-83-83-83-83-83-10	8	0.13	37	3.24	1
	43	73, 83	10-43-43-43-10	5	0.12	25	2.73	1
	53	83	10-53-53-10	4	0.16	4	0.47	1
	71	73	10-71-73-73-10	5	0.12	1	0.20	2
	81	83	10-81-83-10	4	0.12	9	0.75	2
	41	\	10-41-10	3	0.11	1	0.06	2
	13	\	10-13-10	3	0.12	1	0.07	3
Cars	83	53	10-83-83-83-83-83-10	7	0.14	56	3.92	1
	53	43, 83	10-53-53-53-53-10	6	0.18	12	3.93	1
	43	53, 83	10-43-43-10	4	0.12	6	0.80	1
	93	53	93-93-93	3	0.11	5	0.39	1
	91	53	10-91-91-10	4	0.10	7	0.70	2
	51	\	10-51-10	3	0.11	1	0.06	2
	13	\	10-13-10	3	0.12	1	0.07	3
	23	\	10-23-10	3	0.12	1	0.06	3
Trucks-3ax	83	\	10-83-83-83-10	5	0.14	7	0.93	1
	53	83	10-53-53-10	4	0.11	4	0.46	1
	81	\	10-81-81-10	4	0.12	9	0.79	2
	51	53, 83	10-51-83-10	4	0.12	8	0.78	2
	84	\	10-84-10	3	0.13	1	0.06	4
	54	\	10-54-10	3	0.13	1	0.12	4
	52	\	10-52-10	3	0.11	1	0.06	4

Figure 62: The clusters for vans, cars and trucks-3ax

are often visited in the same order across days (e.g. for a single vehicle). More generally, a chain accommodates different sequences of stop locations, but delineates a unique sequence in terms of the classes of the stops. The geographic locations of the stops in a chain may not be the same across days, but the classes (spatial-temporal features) of the stops are identical. For instance, the pattern '10-83-83-83-10' from trucks-3ax reveals that three stops of '83' are often made in a chain between two consecutive stops at work. Each of these stops could be at the same location (thus having the same spatial-temporal features) across different days, while it may also be in different places but belonging to the same classes ('83'). By extracting spatial-temporal features of stops and using these features (stop classes) as basic letters to construct stop-sequences, instead of directly utilizing geographic locations of the stops, stop-sequences across different vehicles can be compared, and patterns that reflect common activity and travel behavior across the vehicles can be identified.

It should be noted that in between each two adjacent stops of a chain, there could be other stops made by the corresponding vehicle. These stops are infrequent (in terms of stop classes) and thus not present in the pattern. Moreover, it was also observed that, the major class (MC) in a chain accounts for most of the stops and the secondary classes (SC) normally have only one (or at most two) instances. For example, over all the patterns (120 in total) in the cluster of MC=('73') for vans, 63 patterns are composed of '73' alone, with the longest pattern as '10-73-73-73-73-73-73-73-73-73-73-73-73-73-10'; while 44 and 11 have only one occurrence of the secondary class '43' and '83' respectively (e.g. '10-73-43-73-73-73-73-73-10' and '10-73-73-73-83-73-10'). In contrast, solely two remaining patterns contain two instances of the secondary classes, including

Category	**1**	**1**	**1**	**1**	**1**	**2**	**2**	**2**	**2**	**2**	**3**	**3**	**4**	**4**	**4**	**4**
Major class (MC)	43	53	73	83	93	41	51	71	81	91	13	23	52	54	74	84
Vans	y	y	y	y		y					y					
Motors	y	y	y	y		y	y		y		y	y				
Trucks-35t	y	y	y	y		y		y			y					
Trucks-3ax		*y*		*y*			*y*						*y*	*y*		*y*
Buses				*y*			*y*							*y*	*y*	*y*
Cars	y	y		y	y		y			y	y	y				

Note: the rows refer to vehicle types, and the columns to categories and major classes. A cell with 'y' indicates the presence of the cluster in the corresponding vehicle type, while a blank cell denotes the otherwise.

Figure 63: Similarities and differences in the clusters across vehicle types

'10-73-73-43-43' and '73-73-83-83-10'. This implies that stops of the same classes (featured with middle-high visit frequency and short travel time) are often planed together during the day.

Category 2 (morning stops): including clusters with MC=('41', '51', '71', '81', '91'). This category comprises patterns in which at least one of the stops is made in the morning (before 10:30am).

Category 3 (infrequent stops): including clusters with MC=('13', '23'). The distinctive feature of this category lies in the low visit frequency of the major stops (FreqPerDay<0.02).

Category 4 (long-travel-time-stops): including clusters with MC=('52', '54', '74', '84'). Category 4 accommodates patterns of stops with long travel time (TravelTime > 60min), marking an important difference from the other categories.

The above four categories characterize activity and travel features of business travel from various aspects, based on which similarities and differences between vehicle types can be delimited (See Figure 63). Vans, motors and trucks-35t share a long chain of (short-middle duration) stops (i.e. '43', '53', '73', '83') (in Category 1), and the longest pattern length is 15, 9 and 8 respectively. These vehicles also expose similarities of regularly travelling to an infrequent location (FreqPerDay<0.02) (in Category 3) for possibly exploring new business opportunities. Likewise, trucks-3ax and buses are also characterized with chains of stops ('83', '53'). Nevertheless, compared to the previous vehicles, the length of these chains is shorter (i.e. the maximum length being 5 for each type) and the duration of these two stops is middle (25min<Duration<140min). Furthermore, these two types have no patterns of infrequent stops, but demonstrate extra patterns featuring long-travel-time-stops (TravelTime>60min) (in Category 4). The remaining type of cars exhibits a high level of deviations from the other types. One outstanding feature of car is having patterns with long duration (Duration>140min) stops ('91', '93'). Moreover, similar to vans, trucks-35t and motors, cars do not have patterns with long-travel-time-stops, implying that most of the car trips are made within an hour.

(4) Travel behavior analysis on a specific company or vehicle

For each vehicle type, the derived clusters along with the sequence support (SuN) form the typical activity-travel profile, based on which travel behavior of a specific company or vehicle can be analyzed. In this process, GPS data of a set of (or a single) sample vehicle(s) (of a certain type) from the study company are first extracted and converted into stop-sequences. Stop-patterns are

Company	Category 1	Category 2	Category 4	Middle SD	Extra patterns
1950	Below (0.18)	Below (0.35)	Normal (1.14)	Normal (0.64)	\
3690	**Above (1.78)**	Normal (1.14)	**Above (2,21)**	**Above (1.78)**	10-62-10 (0.08)
5040	Normal (1.12)	**Above (1.87)**	Normal (0.65)	Normal (1.13)	10-91-10 (0.17)
5940	Normal (0.53)	Normal (0.66)	Normal (0.87)	Normal (0.71)	10-94-10 (0.31)
6060	Normal (0.60)	Normal (0.53)	Below (0.16)	Below (0.39)	10-63-10 (0.13); 10-41-10 (0.1)
6480	Below (0.09)	Below (0.08)	Below (0.05)	Below (0.09)	91-91-91 (0.14); 10-63-10 (0.10) 92-92-92(0.06)

Note: the columns from the left to right denote the companies, Category 1, 2 and 4, middle stop-duration, and extra patterns and the corresponding sequence support which do not fall into any previously-derived common clusters. For trucks-3ax, Category 3 is not present, and all the major classes have middle stop-duration. For the ratios, three levels are used including: below (*<0.5*), normal (*0.5~1.5*), and above (*>1.5*) (in bold).

Figure 64: Ratios on trucks-3ax from six companies

Vehicle	Category 1	Category 2	Category 4	Middle SD	Extra patterns
V1	Below (0.26)	Normal (1.05)	**Above** (3.09)	**Above** (1.70)	10-64-10 (0.24); 10-74-10 (0.24)
V2	Normal (0.58)	Below (0.25)	**Above** (2.22)	Normal (1.17)	10-21-10 (0.16); 10-24-10 (0.16)
V3	**Above** (1.83)	Normal (0.59)	**Above** (1.90)	**Above** (1.51)	92-73-10 (0.27); 10-62-10 (0.14)
V4	Normal (0.90)	Normal (0.91)	Normal (1.49)	Normal (1.15)	10-22-10 (0.18); 10-32-10 (0.18)
V5	**Above** (2.89)	**Above** (1.59)	**Above** (2.58)	**Above** (2.39)	10-71-10 (0.30); 10-21-10 (0.12)
V6	**Above** (2.43)	**Above** (2.23)	**Above** (2.42)	**Above** (2.11)	10-71-10 (0.38); 10-62-10 (0.14)

Note: the columns from the left to right denote the vehicles, Category 1, 2 and 4, middle stop-duration and extra patterns

Figure 65: Ratios on trucks-3ax from six individual vehicles from the company '3690'

then identified and clustered; let SuN' be the obtained sequence support of each cluster. Next, a set of ratios are defined, including R-clu (the ratio between SuN' and SuN of the corresponding cluster), R-cat (the average of R-clu over all clusters in a category), and R-short, R-middle and R-long (the average of R-clu over the clusters that have the major class MC with short, middle and long stop-duration, respectively). These ratios describe the deviations between the sequence support of the sample vehicle(s) and the typical sequence support derived from all vehicles (of the corresponding type) across all companies in the training data. A larger (e.g. >1.5) or smaller (e.g. <0.5) value suggests a higher or lower percentage of stop-sequences of the sample vehicle(s) which match patterns of the corresponding cluster (R-clu), category (R-cat), or level of stop-duration (R-short, R-middle or R-long), than the average percentage of such sequences from a representative vehicle of this type. Figure 64 illustrates the obtained ratios on trucks-3ax from six (transport or construction) companies, while Figure 65 presents the values for six individual vehicles from one ('3690') of the above companies.

From the above two figures, different ratios were noticed, demonstrating variations in activity and travel practice across companies or individual vehicles. From these results, novel insights into travel behavior can be gained, providing a deep understanding of general travel features as well as specific features of the companies (or vehicles). Moreover, these results can be further utilized on driving safety analysis. Due to the inherent nature of the jobs, commercial vehicle drivers are more involved in travelling, leading to more stress and tiredness particularly over long hours of journeys. This increases the risk of crash or other safety-critical events, making safe driving a great challenge for business travel. Research has been carried out to identify significant

factors that are related to driving safety. Apart from drivers' social-economic backgrounds, work and driving characteristics are found as important factors. One of the characteristics concerns driving time; traffic rush hour is considered as a critical timing for driving safety [88]. Driving safety also decreases as the duration of continuous driving gets longer, especially beyond 4 hour [80]. Similarly, the additional time for non-driving activities at stops (e.g. loading-unloading or waiting in queues for trucks) can also increase drivers' fatigue [1]. Moreover, it was found that a proportion of drivers who drive to an unfamiliar place may have difficulties in finding parking locations, leading to a risk of driving as they have to divert attention (e.g. by using mobile phones) in searching for parking [88]. These characteristics affecting driving safety can be reflected by these ratios. For instance, a high value of R-cat for Category 1 (e.g. the company '3690' and vehicles V3, V5 and V6) would suggest a long trip chain with more stops. Likewise, a high value of R-cat for Category 2 (e.g. '5040', V5 and V6), 3, or 4 (e.g. '3690', V1-V3 and V5-V6) indicates more travel in the morning rush hour, more stops that are unfamiliar to the drivers, and more long-time continuous driving, respectively. These trips would lead to longer consecutive time of working and less breaks during the trip chain, more stress from the morning driving, more difficulties in finding parking places, and/or longer continuous driving time. If these trips are performed frequently and become routine travel features of a company (or vehicle), this would potentially increase drivers' fatigue and impair driving performance, leading to high risks of accidents. These ratios would be adopted as additional factors for the analysis and prediction of driving safety, thus adding an extra dimension of travel behavior aspects to the safety research.

3.8 Semantic Enrichment of Trajectory for Cross-Scale Analysis

In summary:

- *Generic question addressed*: Adapting vehicle trajectory simplification process to the external geographical context.
- *Track&Know specific question*: Big mobility data simplification for supporting dashboard visualization (D5.3)
- *Novelty / Advantage over existing methods*: A computationally efficient and effective new trajectory method that can adapt to external geographical context, which provides a different concept perspective from existing geometry-based methods. Parameters are simply for tuning. The method can be embedded as a module into the dashboard workflow.
- *Experiments conducted*:Tested on a TrackKnow Pilot dataset with qualitative and quantitative evaluations.
- *Type of analytics*: Descriptive analysis
- *Automation / TRL*: TRL level 3 (proof-of-concept implemented and tested)
- *Extension to other domains*: The method is applicable for GPS trajectories of other moving objects, e.g., migrating birds and vessels.

3.8.1 A POI-Quadtree-based Variable-Resolution Enrichment Model for Trajectory Simplification

Big GPS trajectory datasets can have redundant spatio-temporal information for applications. The information redundancy introduces extra time on computing. Therefore, trajectory simplification is often a key preprocessing step before the actual data analysis. Many existing simplification methods focus on the geometric information from a trajectory per se. Conversely, methods considering geographic context often fail to provide spatially adaptive simplification or require complex parameter settings to achieve this task. We implemented a novel two-stage adaptive trajectory simplification method, embedding spatial indexing, enrichment, and aggregation in an integrated process. The first stage employs a quadtree for the subdivision depending on the density of geographic context features (e.g., POIs), leading to a variable-resolution representation of the area. The second stage aggregates trajectory waypoints locating in the same quadtree leaf node into a representative point, making the aggregation adapting to the spatial layout of the geographic features in the area of interest. Evaluation with a sample set of the VFI trajectory data shows that the new approach can automatically simplify trajectory segments at variable

compression ratios, with greater simplification in areas with sparse context features (e.g., rural) and less simplification in areas with dense context features (e.g., urban). More importantly, the method can still preserve a variety of inter-trajectory distances between original trajectories and simplified ones, while significantly reducing the computing time. More details are reported in Section 2.2 - 2.4 of D5.3 and [50][51].

3.8.2 The Application of Variable-resolution Enrichment Model for Cross-scale Visual Analytics on Dashboard

Visualizing a big set of raw trajectory waypoints is often challenging as it usually requires high network bandwidth and computational capacity on the browser. The efficiency of the quadtree-based hierarchical structure for the enrichment and simplification of trajectories allows to transfer the simplified big trajectory data to be visualized on a web-based dashboard easily while still preserves the main characteristics of the original trajectory and provides some capacity to interact with the raw waypoints. We implemented a web-mapping based plugin that enables such visualization on our dashboard views (Figure 66). The plugin can adjust spatial details of a trajectory to different spatial scales as end users zoom in and out the maps. The integration of the plugin and the interactive map thus allows multi-scale visual analytics on the trajectories. More details were reported in Section 2.5 of D5.3.

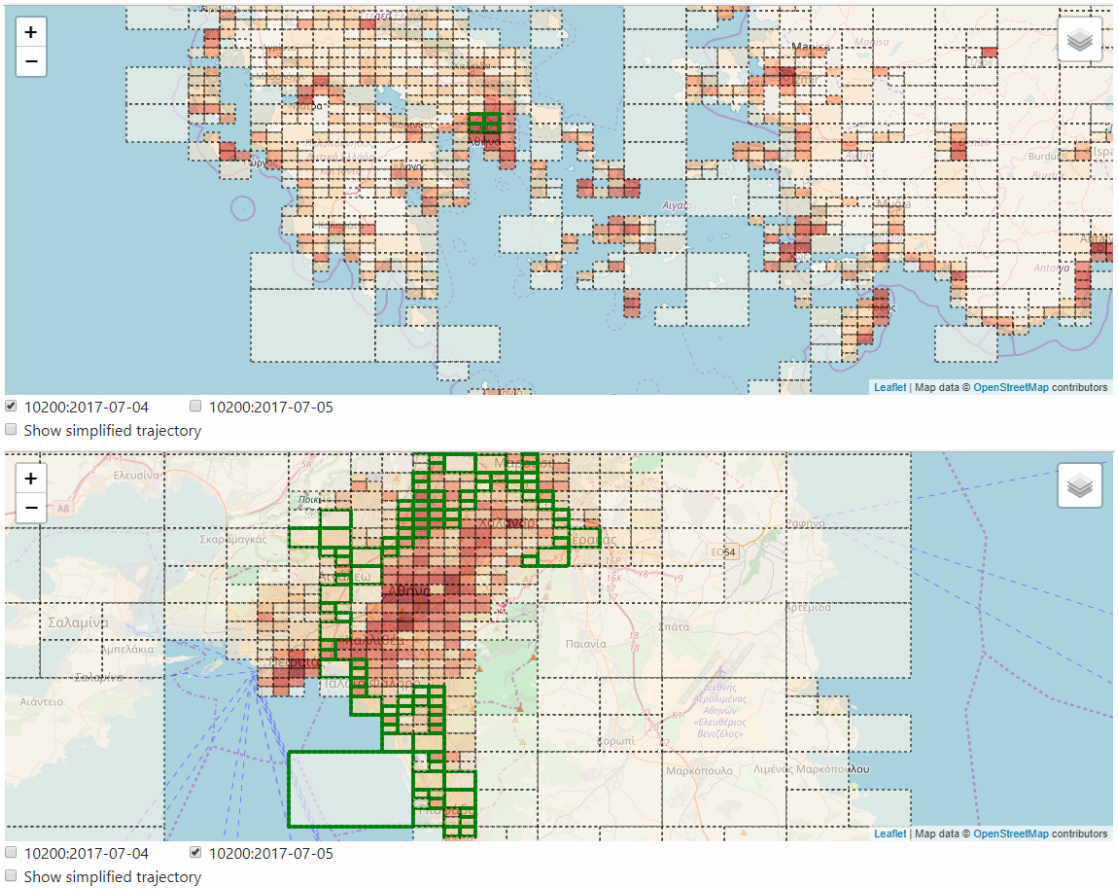

Figure 66: Visualization of a trajectory on top of the POI-quadtree by the plugin. The POI-quadtree nodes that are passed by the trajectory are highlighted. The highlighting is dynamically rendered at different spatial scales.

3.9 Genetic p-Median Solver for Mobility driven Location-Allocation

In summary:

- *Generic question addressed*: What is the best place to locate resources to ensure travel times and kept to a minimum?
- *Track&Know specific question*: Using mobility information can recommendations be made to help optimise location of facilities such as charging stations or clinics?
- *Novelty / Advantage over existing methods*: Open-source, utilises Genetic Algorithm, node level parallelism and Apache Spark parallelism, takes time (along with distance) as a cost.
- *Experiments conducted*: Analysis of Location-Allocation of Medical devices for Healthcare pilot.
- *Type of analytics*: Descriptive analytics
- *Automation / TRL*: 8 (docker container available to reuse, based on previous TRL5 models and methods, results validated against commercial tools)
- *Extension to other domains*: Can be applied to many domains that require location-allocation.

3.9.1 Description of Problem and Approach

Location-allocation problems typical deal with provisioning of resources between facilities based on historic demand. The p-median approach is one such model that aims to minimise the total demand-weighted distance between the demand points and the facilities. This NP-Hard problem aims to locate p facilities to serve n demand, by minimising the total demand-weighted distance between the facilities and the demand. Given the computational complexity of the p-median, several approaches have been proposed to solve problems in polynomial time. Solutions using trees [59] and heuristics (metaheuristics [116], Lagrangian heuristics [33], etc.). Several approaches using genetic algorithms have also been proposed to leverage the power of AI in solving the p-median problem in polynomial time [20][6].

For Track&Know the p-median solver plays an important role in translating mobility information into policy and management recommendations. Plugging into the mobility data processing pipelines of Pilot 1: Insurance and Migration to EV, and Pilot 2: Healthcare Service Optimisation, the p-median solver can provide important business insights for each domain. In the case of migration to electric vehicles, once journey start/end points are identified these can be added to the p-median solver that can recommend location and number of chargers required

in the geographic area (residential stop points are stripped out). Within the healthcare service the p-median solver can provide location and frequency for pop-up clinics, or service capacity recommendations on fixed points within the catchment road network.

3.9.2 Methodology

For Track&Know , the project has adapted the Genetic Algorithm approach to the p-Median solver proposed by Alp and Erkut (2003)[6]. This section covers the tool pipeline and the pre and post-processing, and the next section, 3.9.3 covers the specific implementation of the genetic p-median component.

The current implementation of the p-median module is unconstrained by capacity parameters at the facilities. By default, the placement of facilities is done by dividing the catchment area into a geographic grid. The module can also be invoked and provided with a fixed set of locations for the facilities and then the optimal number of facilities is calculated in a technique defined as conditional p-median.

3.9.2.1 Unconditional p-Median When not providing the solver fixed locations, the module needs to establish a geographic area in which to calculate the p-median. The consumes data from the Track&Know platform using either topics of trajectory data or specific topics that give start-end points of journeys (e.g. patient appointment information). Reading in the data a geographic bounding box is created giving a 0.1-degree buffer on all sides. This creates the catchment area on which the p-median will be performed. Ideally an offline process is required to filter out significant outliers.

The larger the geographic area the more the processing complexity as it would increase the number of grid squares. The smaller the sizes of the grid square the more calculations that need to take place and the more possibilities for the value of p. This is why it is better to start the p-median solver with a smaller geographic catchment area and prune demand points that fall outside this region, (a) and (b) in Figure 67. Further, a larger area with no demand in many grid cells will create a sparse matrix and may also bias the location decisions.

The Track&Know p-Median by default is not restricted to either the P existing facilities, or the n demand points. The catchment area is divided into a grid based on user input parameters. The density of the grid can be adjusted by providing degrees of latitude. Each cell in the grid is a possible location for a facility, (c) in 67. There is a 1 to 1 mapping for facilities to grid squares. This approach allows for transferability to other geographic regions and can deal with unoccupied locations. The resultant output of this stage will be an $m\ x\ n$ matrix with the corner coordinates and the Euclidian centre point for that grid square.

With the catchment area divided into a grid and the demand points pruned, the implementation then calculates the demand for each grid square. This is done by tallying the number for demand points that exist within each cell, by using the geographic location of the demand point and then testing against the corner coordinates of that grid cell. Programmatically this is done

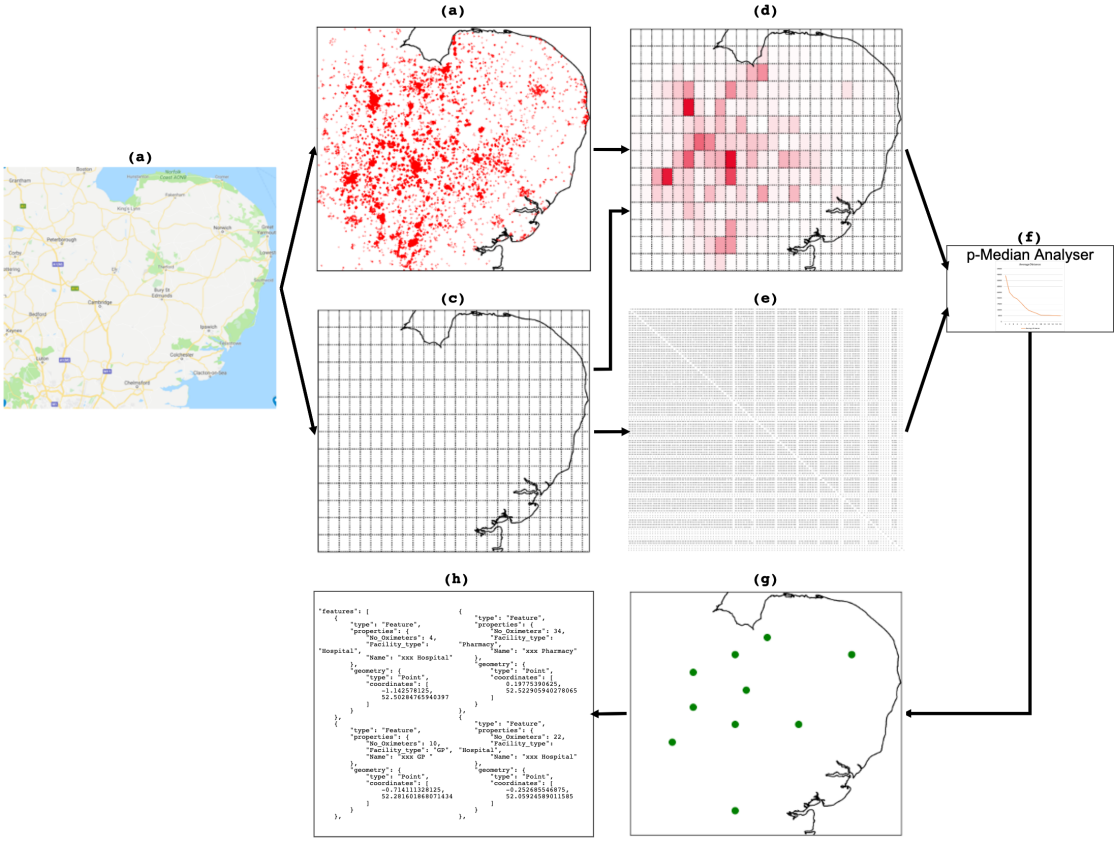

Figure 67: Figure depicting the p-Median workflow. (a) Catchment area definition, (b) mapping >97% of the service users in the catchment, (c) Creating a grid based on user supplied granularity, (d) merging the grid and users to determine demand per grid square, (e) calculating distances between each grid square, (f) a genetic algorithm to calculate the P-optimal locations/grid squares based on distances and demand (g) Resultant output of optimal locations for resource allocation, (h) final GeoJSON output for dashboard integration

by traversing the matrix from (0,0) -> (m,n) by cell by row and once a demand point is *binned* it is not considered when demand for the next cell is calculated. Therefore, if a point exists on the edge of a grid cell then it is only assigned to one cell and is not double counted. This generates an m x n matrix which holds the demand value (Figure 67 (d)).

To calculate the cost between each grid-square a deployment of a customised OpenRouteService (ORS) **??** that is integrated within the Track&Know platform is used. ORS can provide either the road distance (in m) or the journey time (in sec), and these values can be used as costs. The centre of each grid square is used, and ORS is configured to find the closest road within two kilometres of the centre point. Before running the calculate the grid cell centre is checked to be on land using the Python Basemap tool. If the centre is not on land, then in a clockwise manner, starting bottom left, each corner is tested to be on land. The first corner that

does map to land is used and ORS is set to only snap to a road within 350m of the corner. If there is no valid corner point or no possible road for ORS to snap to then the cell is marked off as a non-feasible location. The code to evaluate this matrix is parallelised to utilise multiple threads and cores through the Python Pools framework. The resultant optimisation reduces the execution time from 8 hours to 15 minutes to evaluate a matrix of 1000 x 1000 (1M cells).

This resultant matrix is *'m x n' x 'm x n'*, with each cell mapping across all other cells (Figure 67 (e)). Typically, in this configuration the matrix is meant to reflect over the diagonal, however, as the distance/time value is calculated using the underlying road network the mirrored cell will not contain the same value. Due to roundabouts and one-way systems, the journey from Cell 1 to Cell n will not equal the journey back from Cell n to Cell 1.

Using the costs and the demand matrices, step (f) performs the p-median, optimising by minimising distance or time of journey based on demand (see: Section 3.9.3). The simulation runs several times each for p from 1 to 'X' as specified by the user.

The resultant output is a set of GeoJSON's for each value of p. Contained within the output is the total distance travelled or time taken commuting by all demand points, and the recommendation of locations for the p facilities. The distance or time values when plotted should create an elbow curve where a point occurs after which inserting a new facility does not significantly improve distance/time value. The user can use this graph to decide which value of p is most favourable.

The output GeoJSON's are read into the Track&Know dashboard and visualised both on a map and using a graph. Other metrics such as max distance travelled by a single demand point, average distance, and total distance are also given to the user.

3.9.2.2 Conditional p-Median Steps (b) – (d) can be replaced with a set of fixed facilities (F) for the p-Median function to calculate the optimal p out of F. The demand is calculated by tallying demand points based on proximity to facilities. The distance between the demand point, d, is calculated against all F facilities and the minimum value (closest facility) is used to assign demand. The distance or time optimisation then takes place by identifying distances between each p.

The remaining pipeline of the tool is the same as for unconditional p-Median.

3.9.3 Implementation

3.9.3.1 p-Median Function The formulation of the p-median problem utilised by the Track&Know project is the one by ReVelle and Swain [154]. This is an unconstrained formulation of the problem as it does not take facility capacity into account when making location-allocation decisions. The other consideration is the 1-to-1 mapping of a demand point to a facility. i.e. a demand point can only be served by exactly one facility. Equation (1) is the objective function of the model, where the total demand-weighted distance between the demand points and facilities needs to be minimised.

$$min \sum_{i=1}^{n} w_i d_{ij} x_{ij} \tag{22}$$

$$s.t. \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i, \tag{23}$$

$$x_{ij} \leq y_i \qquad \forall i, j, \tag{24}$$

$$\sum_{j=1}^{n} y_j = p, \tag{25}$$

$$x_{ij} = 0 \, or \, 1 \qquad \forall i, j, \tag{26}$$

$$y_j = 0 \, or 1 \qquad \forall j, \tag{27}$$

where

$$n = total \; number \; of \; demand \; points \tag{28}$$

$$x_{ij} = \begin{cases} 1, & \text{if point } i \text{ is assigned to facility located at point } j, \\ 0, & otherwise, \end{cases} \tag{29}$$

$$y_j = \begin{cases} 1, & \text{if a facility is located at point } j, \\ 0, & otherwise, \end{cases} \tag{30}$$

$$w_i = demand \; at \; point \; i, \tag{31}$$

$$d_{ij} = travel \; distance \; between \; points \; i \; and \; j, \tag{32}$$

$$p = number \; of \; facilities \; to \; be \; located. \tag{33}$$

3.9.3.2 Genetic Solver While many heuristic approaches to solving the p-Median problem have been proposed, the genetic algorithm proposed by Alp and Erkut (2003)[6], has been found through experimentation to be efficient and whose results are comparable to commercial location-allocation solvers offered in packages such as ArcGIS.

To setup the genetic algorithm, we use the objective function (1) as the fitness function for the evaluation of the quality of a chromosome. The genes in the chromosome correspond to the facility number/grid cell number, e.g. for a 10 by 10 grid the genes would be labeled from 1 to 100. The length of the chromosome will be governed by the value of p, e.g. for a 4-median problem the chromosome would have 4 genes. [10, 23, 5, 97] is a valid chromosome representing solutions in grid cells' 5, 10, 23, 97.

$$P(n, p) = max \left\{ 2, \left\lceil \frac{n}{100} . \frac{ln(S)}{d} \right\rceil \right\} d \tag{34}$$

To start the solution the genetic algorithm needs a population size that remains fixed. [154] provide a formula (13) to evaluate the population size P, which is dependant on S (the number of all possible solutions) and d (integer value of demand density). The population size is large enough to encapsulate all the genes (i.e. every location). The population is then initialised by sequentially added each gene to the chromosome. So in a 4-median solution the first chromosome would be [1, 2, 3, 4], and the second would be [5, 6, 7, 8] etc. The fitness of each chromosome is calculated and this value is held to be compared against the generated chromosomes.

When the simulation runs, two parents are randomly selected from the population and their chromosomes merged. The resultant 8-gene chromosome is not valid and therefore 4 genes have to be dropped. The greedy deletion heuristic does not delete genes that exist in both parents. Once the chromosome is of length p, the fitness is calculated and compared to the fitness of the population. If the fitness is better than the worst performing chromosome in the population, then the new chromosome replaces the worst one. This process is repeated until one solution remains.

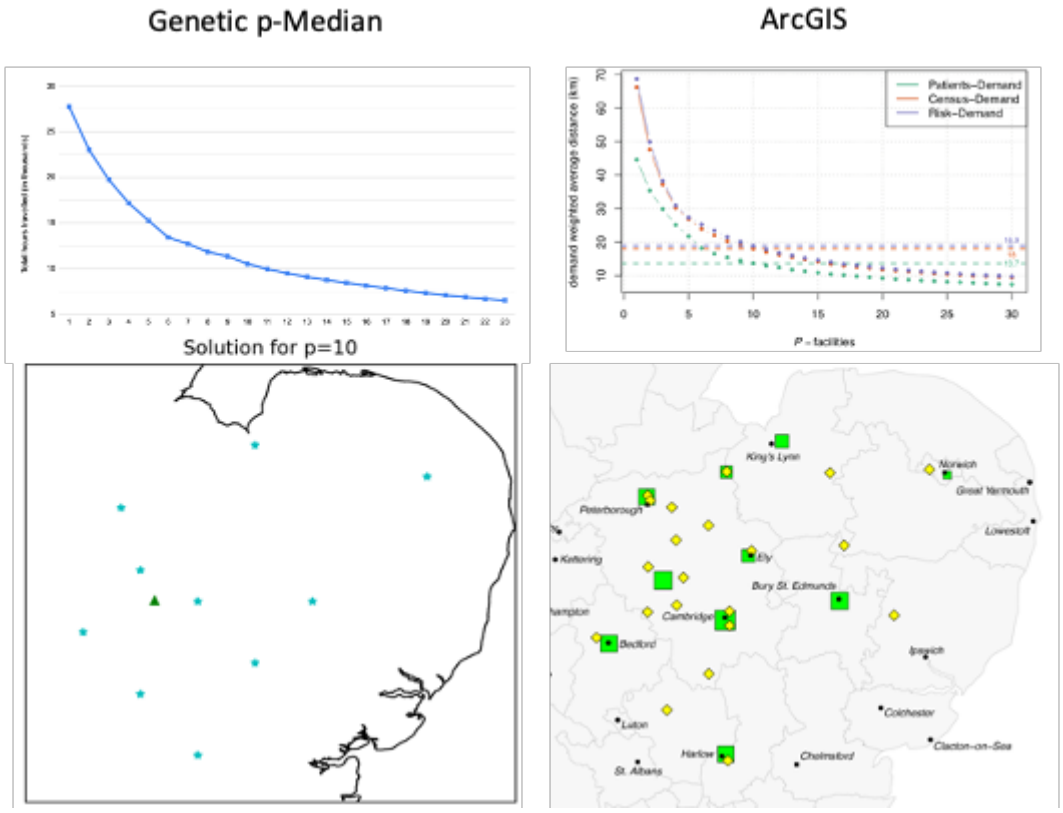

Figure 68: Result of the Track&Know Genetic p-Median solver and a comparison to a commercial solution

3.9.3.3 Code Parallelisation To parallelise the implementation of p-Median to make it Big Data ready the implementation of the genetic algorithm conforms to the solution presented by Maqbool et. al [114]. The Scalable Genetic Algorithm (S-GA) implementation that has been developed under the Boost 4.0, LAMBDA, SLIPO, and QROWD projects, provides a parallelisation strategy that utilises the Apache Spark framework. This framework is also utilised within the Track&Know Platform (see: D2.1).

The performance results of the code, will be reported in subsequent deliverables.

3.9.4 p-Median Application

The p-Median tool will be used in the Healthcare pilot to provide mobility aware location-allocation solutions for medical facility placement, and the Electric Mobility part of the insurance pilot to find optimal charge point locations. Initial tests have been run against the healthcare datasets using both the p-median solver and a commercial tool ArcGIS (Figure 68). While the p-medians are slightly different in the exact placement of facilities, both solutions identified specific key hot-spots and were in agreement in terms of optimised numbers. The differences recorded here are due to ArcGIS using a conditional p-median approach. Further testing will be carried out as part of pilot activities and reported in D6.2, D6.3 and D6.6, respectively.

4 Conclusions

In this deliverable D4.1, the BDA Toolbox is described in detail regarding all the aspects of the development. Each individual component is presented separately, including the formalization of the problem it addresses, the design of the functionality provided, as well as the experimental evaluation of its implementation.

Although autonomous, the presentation of the BDA components show that the Toolbox is indeed a collection of closely collaborating modules that provide top-level BDA functionalities, ranging from customized pre-processing and data restoration & transformation per-component to trajectory analytics & clustering, discovery of mobility networks & personalized trips, hotspot analysis, driver behaviour profiling, traffic flow dynamics, etc. Hence, the BDA functional requirements are met and at the same time the Toolkit is versatile and modular enough that it can be employed to a much wider range of challenges than what is included in the scope of Track&Know.

5 Annex I: Ethics report

A. PERSONAL DATA

1. Is personal data going to be processed for the completion of this deliverable?
   (a) If "yes", do they refer only to individuals connected to project partners? Or to third parties as well?

   We have used anonymized data from the data providers in the project, corresponding to the three pilots. The anonymization process took place prior to data release for usage in Track&Know.

2. Are "special categories of personal data" going to be processed for this deliverable? (whereby these include personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, and trade union membership, as well as, genetic data, biometric data, data concerning health or data concerning a natural person's sex life or sexual orientation) **No.**

3. Has the consent of the individuals concerned been acquired prior to the processing of their personal data?
   (a) If "yes", based on the Project's Consent Form? On a different legal basis?

   Yes, the partners that act as data providers in the project have acquired consent from individuals prior to data processing. In the case of personal data, a consent form has been used. In case of data coming from one of the clients of a data provider, appropriate agreements have been made about the usage of such data, always appropriately anonymized.

4. In the event of processing of personal data, is the processing:
   (a) "Fair and lawful", meaning executed in a fair manner and following consent of the individuals concerned? **Yes.**
   (b) Performed for a specific (project-related) cause only? **The usage of any data within the Task 4.1 and Deliverable D4.1 concerned only implementation and testing purposes of the developed tools. For the fulfilment of these tasks appropriately anonymized and de-identified data samples were provided by partners.**
   (c) Executed on the basis of the principle of proportionality (meaning that only data that are necessary for the processing purposes are being processed)? **Yes.**
   (d) Based on high-quality personal data? **(see previous)**

5. Are all lawful requirements for the processing of the data (for example, notification of the competent Data Protection Authority(s), if applicable) adhered to? **Not applicable.**

6. Have individuals been made aware of their rights (particularly the rights to access, rectify and delete the data)? **Yes, the partners that act as data providers in the projecthave acquired consent from individuals prior to data processing, and took careof informing them of their rights.**

B. DATA SECURITY

1. Have proportionate security measures been undertaken for protection of the data, taking into account project requirements and the nature of the data? **Yes.**

   (a) Brief description of such measures (including physical-world measures, if any)

   There have been a number of security measures in place including: (a) remote access to the machines used only by SSH, (b) data resides on the platform only during the execution of experiments, (c) access to data storage and data feeds require authentication.

2. Is there a data breach notification policy in place within your organisation? **Yes.**

C. DATA TRANSFERS

1. Are personal data transfers beyond project partners going to take place for this deliverable? **No.**

   (a) If "yes", do these include transfers to third (non-EU) countries?

2. Are personal data transfers to public authorities going to take place for this deliverable? **No.**

   (a) Do any state authorities have direct or indirect access to personal data processed for this deliverable?

3. Taking into account that the Project Coordinator is the "controller" of the processing and that all other project partners involved in this deliverable are "processors" within the same contexts, are there any other personal data processing roles attributed to any third parties for this deliverable? **No.**

D. ETHICS AND RELATED ISSUES

1. Are personal data of children going to be processed for this deliverable? **No.**

2. Is profiling in any way enabled or facilitated for this deliverable? **Yes, however GDPR compliance measures are applied.**

3. Are automated-decisions made or enabled for this deliverable? **No.**

4. Have partners for this deliverable taken into consideration system architectures of privacy by design and/or privacy by default, as appropriate? **Yes.**

5. Have partners for this deliverable taken into consideration gender equality policies? **Not applicable.**

6. Have partners for this deliverable taken into consideration confidentiality of the data requirements? **Yes.**

References

[1] Williamson A. and Friswell R. The effect of external non-driving factors, payment type and waiting and queuing on fatigue in long distance trucking. *Accident Analysis and Prevention*, 58:26–34, 2013.

[2] Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. Subtrajectory clustering: Models and algorithms. In *PODS*, pages 75–87, 2018.

[3] Louai Alarabi and Mohamed F. Mokbel. A demonstration of st-hadoop: A mapreduce framework for big spatio-temporal data. *PVLDB*, 10(12):1961–1964, 2017.

[4] Louai Alarabi, Mohamed F. Mokbel, and Mashaal Musleh. St-hadoop: A mapreduce framework for spatio-temporal data. In *Proceedings of the 15th International Symposium on Spatial and Temporal Databases, SSTD*, pages 84–104, 2017.

[5] A. Alexandridis, E. Chondrodima, and H. Sarimveis. Radial basis function network training using a nonsymmetric partition of the input space and particle swarm optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 24(2):219–230, 2013.

[6] Osman Alp, Erhan Erkut, and Zvi Drezner. An efficient genetic algorithm for the p-median problem. *Annals of Operations research*, 122(1-4):21–42, 2003.

[7] F. Altché and A. de La Fortelle. An lstm network for highway trajectory prediction. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 353–359, Oct 2017.

[8] Gamallo Alvaro, Gonzalez and Fraile-Ardanuy. Estimation of electric vehicles' consumption based on their mobility. Section 3.3.3. of Deliverable D6.1 of the DataSIM Project, 2013.

[9] Theodoros Anagnostopoulos, Christos Anagnostopoulos, and Stathes Hadjiefthymiades. Mobility prediction based on machine learning. In *12th IEEE International Conference on Mobile Data Management, MDM 2011, Luleå, Sweden, June 6-9, 2011, Volume 2*, pages 27–30, 2011.

[10] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: ordering points to identify the clustering structure. In *SIGMOD*, pages 49–60, 1999.

[11] European Automobile Manufacturers Association. Acea report vehicles in use europe. 2019.

[12] Samet Ayhan and Hanan Samet. Aircraft trajectory prediction made easy with predictive analytics. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 21–30, 2016.

[13] Samet Ayhan and Hanan Samet. Time series clustering of weather observations in predicting climb phase of aircraft trajectories. In *Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS@SIGSPATIAL 2016, Burlingame, California, USA, October 31 - November 3, 2016*, pages 25–30, 2016.

[14] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, 140:112896, 2020.

[15] Kyle Banker, Peter Bakkum, Shaun Verchand, Douglas Garrett, and Tim Hawkins. *MongoDB in Action*. Manning, 2014.

[16] A. Bender, G. Agamennoni, J. Ward, S. Worrall, and E. Nebot. An unsupervised approach for inferring driver behavior from naturalistic driving data. *IEEE Trans. Intell. Transp. Syst.*, 16(6), 2015.

[17] L. Bergasa, J. Nuevo, M. Sotelo, R. Barea, and M. Lopez. Real-time systemformonitoring driver vigilance. *IEEE Trans. on Intell. Trans. Sys.*, 7(1), 2006.

[18] Ella Bingham. Finding segmentations of sequences. In *Inductive Databases and Constraint-Based Data Mining*, pages 177–197. Springer, 2010.

[19] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[20] Burcin Bozkaya, Jianjun Zhang, and Erhan Erkut. An efficient genetic algorithm for the p-median problem. *Facility location: Applications and theory*, pages 179–205, 2002.

[21] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, page 853–864. VLDB Endowment, 2005.

[22] Ronald Bremer. *Outliers in statistical data*. Taylor & Francis, 1995.

[23] Maike Buchin et al. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In *SIGSPATIAL*, pages 202–211. ACM, 2010.

[24] Harmen J Bussemaker et al. Regulatory element detection using a probabilistic segmentation model. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, pages 67–74, 2000.

[25] Huiping Cao, Nikos Mamoulis, and David W. Cheung. Discovery of collocation episodes in spatiotemporal data. In *Proceedings of the 6th IEEE International Conference on Data Mining, ICDM*, pages 823–827, 2006.

[26] G. Castignani, T. Derrmann, R. Frank, and T. Engel. Driver behavior profiling using smartphones: A low-cost platform for driver monitoring. *IEEE Intell. Transp. Syst. Mag.*, 7(1), 2015.

[27] Michelangelo Ceci, Annalisa Appice, and Donato Malerba. Time-slice density estimation for semantic-based tourist destination suggestion. In *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, pages 1107–1108, 2010.

[28] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *Proceedings of the 27th International Conference on Data Engineering, ICDE*, pages 900–911, 2011.

[29] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[30] Kristina Chodorow. *MongoDB: The Definitive Guide*. O'Reilly Media, Inc., 2013.

[31] Seongjin Choi, Jiwon Kim, and Hwasoo Yeo. Attention-based recurrent neural network for urban vehicle trajectory prediction. *Procedia Computer Science*, 151:327 – 334, 2019. The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops.

[32] Christophe Claramunt, Cyril Ray, Elena Camossi, Anne-Laure Jousselme, Melita Hadzagic, Gennady L. Andrienko, Natalia V. Andrienko, Yannis Theodoridis, George A. Vouros, and Loïc Salmon. Maritime data integration and analysis: recent progress and research challenges. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT*, pages 192–197, 2017.

[33] Mark S Daskin. *Network and discrete location: models, algorithms, and applications*. John Wiley & Sons, 2 edition, 2013.

[34] Ze Deng, Yangyang Hu, Mao Zhu, Xiaohui Huang, and Bo Du. A scalable and fast OPTICS for clustering trajectory big data. *Cluster Computing*, 18(2):549–562, 2015.

[35] Xin Ding, Lu Chen, Yunjun Gao, Christian S. Jensen, and Hujun Bao. Ultraman: A unified platform for big trajectory data management and analytics. *PVLDB*, 11(7):787–799, 2018.

[36] Christos Doulkeridis, Akrivi Vlachou, Dimitris Mpestas, and Nikos Mamoulis. Parallel and distributed processing of spatial preference queries using keywords. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT*, pages 318–329, 2017.

[37] Ahmed Eldawy and Mohamed F. Mokbel. The era of big spatial data: A survey. *Foundations and Trends in Databases*, 6(3-4):163–273, 2016.

[38] A Ellison, S Greaves, and M Bliemer. Driver behaviour profiles for road safety analysis. *Accident Analysis and Prevention*, 76:118–132, 2015.

[39] J. Engelbrecht, M. Booysen, J. van Rooyen, and F. Bruwer. Survey of smartphone-based sensing in vehicles for intelligent transportation system applications. *IET Intell. Transp. Syst.*, 9(10), 2015.

[40] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[41] Mohammad Etemad et al. A trajectory segmentation algorithm based on interpolation-based change detection strategies. In *EDBT/ICDT Workshops*, 2019.

[42] Liu F., Janssens D., and Wets G. Identifying business activity-travel patterns based on GPS data. *Paper being drafted*, 2020.

[43] Qi Fan, Dongxiang Zhang, Huayu Wu, and Kian-Lee Tan. A general and parallel platform for mining co-movement patterns over large-scale trajectories. *PVLDB*, 10(4):313–324, 2016.

[44] X. Fan, L. Guo, N. Han, Y. Wang, J. Shi, and Y. Yuan. A deep learning approach for next location prediction. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 69–74, May 2018.

[45] Yixiang Fang, Reynold Cheng, Wenbin Tang, Silviu Maniu, and Xuan S. Yang. Scalable algorithms for nearest-neighbor joins on big trajectory data. *IEEE Transactions on Knowledge and Data Engineering TKDE*, 28(3):785–800, 2016.

[46] F. Feng, S. Bao, J.R. Sayer, C. Flannagan, M. Manser, and R. Wunderlich. Can vehicle longitudinal jerk be used to identify aggressive drivers? an examination using naturalistic driving data. *Accid. Anal. Prev.*, 104, 2017.

[47] Marta Fort, Joan Antoni Sellarès, and Nacho Valladares. A parallel gpu-based approach for reporting flock patterns. *IJGIS*, 28(9):1877–1903, 2014.

[48] Elias Frentzos, Kostas Gratsias, and Yannis Theodoridis. Index-based most similar trajectory search. In *ICDE*, pages 816–825, 2007.

[49] F Fritsch and R Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17:238–246, 1980.

[50] Cheng Fu, Haosheng Huang, and Robert Weibel. Adaptive Simplification of GPS Trajectories with Geographic Context. *International Journal of Geographical Information Science*, 2020 in press.

[51] Cheng Fu and Robert Weibel. Cross-scale Spatial Enrichment of Trajectories for Speeding Up Similarity Computing A Theory of Cross-scale Analysis. In Georg Gartner and Haosheng Huang, editors, *LBS 2019 : Adjunct Proceedings of the 15th International Conference on Location-Based Services*, pages 135–140, Vienna, Austria, 2019.

[52] H Georgiou, P Petrou, Pelekis N, and Y Theodoridis. Unsupervised driver behaviour profiling using sparse gps data for online trajectory analytics. In *(submitted)*, 2020.

[53] Harris Georgiou, Sophia Karagiorgou, Yannis Kontoulis, Nikos Pelekis, Petros Petrou, David Scarlatti, and Yannis Theodoridis. Moving Objects Analytics: Survey on Future Location & Trajectory Prediction Methods. *arXiv e-prints*, page arXiv:1807.04639, Jul 2018.

[54] Harris Georgiou, Nikos Pelekis, Stylianos Sideridis, David Scarlatti, and Yannis Theodoridis. Semantic-aware aircraft trajectory prediction using flight plans. *International Journal of Data Science and Analytics*, pages 1–14, March 2019.

[55] Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Comput.*, 12(10):2451–2471, October 2000.

[56] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining, SIGKDD*, pages 330–339, 2007.

[57] Gyözö Gidófalvi and Fang Dong. When and where next: individual mobility prediction. In *ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS 2012*, pages 57–64, 2012.

[58] C. Goh, J. Dauwels, N. Mitrovic, M. Asif, A. Oran, and P. Jaillet. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *15th IEEE Intl. Conf. Intell. Transp. Sys. (ICITS)*, 2012.

[59] AJ Goldman. Optimal center location in simple networks. *Transportation science*, 5(2):212–221, 1971.

[60] João Bártolo Gomes, Clifton Phua, and Shonali Krishnaswamy. Where will you go? mobile data mining for next place prediction. In *Data Warehousing and Knowledge Discovery - 15th International Conference, DaWaK 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings*, pages 146–158, 2013.

[61] J. Goncalves, J.S. Goncalves, R. Rossetti, and C. Olaverri-Monreal. Smartphone sensor platform to study traffic conditions and assess driving performance. In *Proc. 17th Int. IEEE Conf. Intelligent Transportation Systems (ITSC)*, 2014.

[62] Joachim Gudmundsson, Marc J. van Kreveld, and Frank Staals. Algorithms for hotspot computation on trajectory data. In *Proceedings of the 21st International Conference on Advances in Geographic Information Systems, SIGSPATIAL*, pages 134–143, 2013.

[63] Riccardo Guidotti et al. Never drive alone: Boosting carpooling with network analysis. *IS*, 64:237–257, 2017.

[64] Riccardo Guidotti et al. There's a path for everyone: A data-driven personal model reproducing mobility agendas. In *DSAA*, pages 303–312. IEEE, 2017.

[65] Riccardo Guidotti and Mothers. Retrieving points of interest from human systematic movements. In *SEFM*, pages 294–308. Springer, 2014.

[66] Riccardo Guidotti, Roberto Trasarti, and Mirco Nanni. Tosca: two-steps clustering algorithm for personal locations detection. In *SIGSPATIAL*, page 38. ACM, 2015.

[67] F. Guo, S. Klauer, J. Hankey, and T. Dingus. Near crashes as crash surrogate for naturalistic driving studies. *Transp. Res. Rec.*, 2147, 2010.

[68] Sini Guo et al. Gps trajectory data segmentation based on probabilistic logic. *International Journal of Approximate Reasoning*, 103:227–247, 2018.

[69] Stefan Hagedorn and Timo Räth. Efficient spatio-temporal event processing with STARK. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT*, pages 570–573, 2017.

[70] S. Haykin. *Adaptive Filter Theory (3rd/Ed.)*. Prentice-Hall, 1996.

[71] I. Hazan and A. Shabtai. Improving grid-based location prediction algorithms by speed and direction based boosting. *IEEE Access*, 7:21211–21219, 2019.

[72] B. Higgs and M. Abbas. Segmentation and clustering of car-following behavior: Recognition of driving patterns. *IEEE Trans. Intell. Transp. Syst.*, 16(1), 2015.

[73] Johan Himberg et al. Time series segmentation for context recognition in mobile devices. In *ICDM*, pages 203–210. IEEE, 2001.

[74] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[75] J. Hong, B. Margines, and A. Dey. A smartphone-based sensing platform to model aggressive driving behaviors. In *Proc. 32nd Annu. ACM Conf. Human Factors in Computing Systems (CHI)*, 2014.

[76] Liang Hong, Yu Zheng, Duncan Yung, Jingbo Shang, and Lei Zou. Detecting urban black holes based on human mobility data. In *Proceedings of the 23rd International Conference on Advances in Geographic Information Systems SIGSPATIAL*, pages 35:1–35:10, 2015.

[77] L. Hou, L. Xin, S. E. Li, B. Cheng, and W. Wang. Interactive trajectory prediction of surrounding road users for autonomous driving using structural-lstm network. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2019.

[78] Chunchun Hu, Xionghua Kang, Nianxue Luo, and Qiansheng Zhao. Parallel clustering of big data of spatio-temporal trajectory. In *ICNC*, pages 769–774, 2015.

[79] Chih-Chieh Hung, Wen-Chih Peng, and Wang-Chien Lee. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *VLDB J.*, 24(2):169–192, 2015.

[80] Claveria J.B., Hernandez S., Anderson J.C., and Jessup E.L. Understanding truck driver behavior with respect to cell phone use and vehicle operation. *Transportation Research Part F*, 65:389–401, 2019.

[81] Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. A hybrid prediction model for moving objects. In *ICDE*, pages 70–79. IEEE, 2008.

[82] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, and Christian S. Jensen. Path prediction and predictive range querying in road network databases. *VLDB J.*, 19(4):585–602, 2010.

[83] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *PVLDB*, 1(1):1068–1080, 2008.

[84] Yantao Jia, Yuanzhuo Wang, Xiaolong Jin, and Xueqi Cheng. Location prediction: A temporal-spatial bayesian model. *ACM TIST*, 7(3):31:1–31:25, 2016.

[85] Peiquan Jin, Jiang Du, Chuanglin Huang, Shouhong Wan, and Lihua Yue. Detecting hotspots from trajectory data in indoor spaces. In *Proceedings of the 20th International Conference on Database Systems for Advanced Applications, DASFAA, Part I*, pages 209–225, 2015.

[86] Amílcar Soares Júnior et al. Grasp-uts: an algorithm for unsupervised trajectory segmentation. *International Journal of Geographical Information Science*, 29(1):46–68, 2015.

[87] Amílcar Soares Júnior et al. A semi-supervised approach for the semantic segmentation of trajectories. In *19th IEEE International Conference on Mobile Data Management (MDM)*, pages 145–154, 2018.

[88] Mahajan K., Nagendra R.V., Kumar A., Choudhary A., and Choudhary P. Effects of driver work-rest patterns, lifestyle and payment incentives on long-haul truck driver sleepiness. *Transportation Research Part F*, 60:366–382, 2019.

[89] D Kahaner, C Moler, and S Nash. *Numerical Methods and Software*. Prentice Hall, Upper Saddle River, NJ, USA, 1988.

[90] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *SSTD*, pages 364–381, 2005.

[91] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404, Oct 2017.

[92] Henrik Klessig, Vinay Suryaprakash, Oliver Blume, Albrecht J. Fehske, and Gerhard Fettweis. A framework enabling spatial analysis of mobile traffic hot spots. *IEEE Wireless Commun. Letters*, 3(5):537–540, 2014.

[93] N. Koutroumanis, G. Santipantakis, A. Glenis, C. Doulkeridis, and G. Vouros. Integration of mobility data with weather information. In *Proc. EDBT/ICDT 2019 Joint Conference (EDBT/ICDT)*, 2019.

[94] Dheeraj Kumar, Huayu Wu, Sutharshan Rajasegarar, Christopher Leckie, Shonali Krishnaswamy, and Marimuthu Palaniswami. Fast and scalable big data trajectory clustering for understanding urban mobility. *IEEE Trans. Intelligent Transportation Systems*, 19(11):3709–3722, 2018.

[95] Patrick Laube, Stephan Imfeld, and Robert Weibel. Discovering relative motion patterns in groups of moving point objects. *IJGIS*, 19(6):639–668, 2005.

[96] Victor Lavrenko et al. Mining of concurrent text and time series. In *KDD Workshop on Text Mining*, volume 2000, pages 37–44, 2000.

[97] Jae-Gil Lee et al. Trajectory clustering: A partition-and-group framework. In *ACM SIGMOD*, page 593–604. ACM, 2007.

[98] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD*, pages 593–604, 2007.

[99] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal*, 1, 07 2014.

[100] Po-Ruey Lei, Tsu-Jou Shen, Wen-Chih Peng, and Ing-Jiunn Su. Exploring spatial-temporal trajectory model for location prediction. In *12th IEEE International Conference on Mobile Data Management, MDM 2011, Luleå, Sweden, June 6-9, 2011, Volume 1*, pages 58–67, 2011.

[101] Luis Leiva and Enrique Vidal. Warped k-means: An algorithm to cluster sequentially-distributed data. *Information Sciences*, 237:196–210, 07 2013.

[102] Wentian Li. Dna segmentation as a model selection process. In *Proceedings of the Fifth Annual International Conference on Computational Biology*, RECOMB '01, page 204–210. ACM, 2001.

[103] Yuxuan Li, James Bailey, and Lars Kulik. Efficient mining of platoon patterns in trajectory databases. *Data Knowl. Eng.*, 100:167–187, 2015.

[104] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *PVLDB*, 3(1):723–734, 2010.

[105] N. Lin, C. Zong, M. Tomizuka, P. Song, Z. Zhang, and G. Li. An overview on study of identification of driver behavior characteristics for automotive control. *Math. Probl. Eng.*, 2014, 2014.

[106] T. Lofstedt, P. Brynolfsson, T. Asklund, T. Nyholm, and A. Garpebring. Gray-level invariant haralick texture features. *PLoS ONE*, 14(2), 2019.

[107] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, page 352–361, New York, NY, USA, 2009. Association for Computing Machinery.

[108] Eric Hsueh-Chan Lu, Vincent S. Tseng, and Philip S. Yu. Mining cluster-based temporal mobile sequential patterns in location-based service environments. *IEEE Trans. Knowl. Data Eng.*, 23(6):914–927, 2011.

[109] Jonas Lukasczyk, Ross Maciejewski, Christoph Garth, and Hans Hagen. Understanding hotspots: A topological visual analytics approach. In *Proceedings of the 23rd International Conference on Advances in Geographic Information Systems SIGSPATIAL*, pages 36:1–36:10, 2015.

[110] Y Ma and et.al. A comparative study of aggressive driving behavior recognition algorithms based on vehicle motion data. *IEEE Access*, 7:8028–8038, 2018.

[111] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *AAAI*, 2018.

[112] Zhiming Ma, Mengfei Yao, Tao Hong, and Bo Li. Aircraft surface trajectory prediction method based on LSTM with attenuated memory window. *Journal of Physics: Conference Series*, 1215:012003, may 2019.

[113] Gabor Makrai. Efficient method for large-scale spatio-temporal hotspot analysis. In *Proceedings of the 24th ACM International Conference on Advances in Geographic Information Systems, SIGSPATIAL*, 2016.

[114] Fahad Maqbool, Saad Razzaq, Jens Lehmann, and Hajira Jabeen. Scalable distributed genetic algorithm using apache spark (s-ga). In *International Conference on Intelligent Computing*, pages 424–435. Springer, 2019.

[115] G. Meiring and H. Myburgh. A review of intelligent driving style analysis systems and related artificial intelligence algorithms. *Sensors*, 15(12), 2015.

[116] Nenad Mladenović, Jack Brimberg, Pierre Hansen, and José A Moreno-Pérez. The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, 179(3):927–939, 2007.

[117] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *ACM SIGKDD*, pages 637–646, 2009.

[118] P.A. Moran. Notes on continuous stochastic phenomena. *Biometrika*, 37(1):17–23, 1950.

[119] Brendan Morris and Mohan M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *IEEE CVPR*, pages 312–319, 2009.

[120] Mikolaj Morzy. Mining frequent trajectories of moving objects for location prediction. In *Machine Learning and Data Mining in Pattern Recognition, 5th International Conference, MLDM 2007, Leipzig, Germany, July 18-20, 2007, Proceedings*, pages 667–680, 2007.

[121] Roger Moussalli, Ildar Absalyamov, Marcos R. Vieira, Walid A. Najjar, and Vassilis J. Tsotras. High performance FPGA and GPU complex pattern matching over spatio-temporal streams. *GeoInformatica*, 19(2):405–434, 2015.

[122] Attila M. Nagy and Vilmos Simon. Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing*, 50:148 – 163, 2018.

[123] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.*, 27(3):267–289, 2006.

[124] Pascal Neis and Alexander Zipf. Openrouteservice. org is three times "open": Combining opensource, openls and openstreetmaps. *GIS Research UK (GISRUK 08). Manchester*, 2008.

[125] Panagiotis Nikitopoulos, Aris-Iakovos Paraskevopoulos, Christos Doulkeridis, Nikos Pelekis, and Yannis Theodoridis. BigCAB: Distributed hot spot analysis over big spatio-temporal data using Apache Spark. In *Proceedings of the 24th ACM International Conference on Advances in Geographic Information Systems, SIGSPATIAL*, 2016.

[126] Panagiotis Nikitopoulos, Aris-Iakovos Paraskevopoulos, Christos Doulkeridis, Nikos Pelekis, and Yannis Theodoridis. Hot spot analysis over big trajectory data. In Naoki Abe, Huan Liu, Calton Pu, Xiaohua Hu, Nesreen K. Ahmed, Mu Qiao, Yang Song, Donald

Kossmann, Bing Liu, Kisung Lee, Jiliang Tang, Jingrui He, and Jeffrey S. Saltz, editors, *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 761–770. IEEE, 2018.

[127] Masaaki Nishino, Yukihiro Nakamura, Takashi Yagi, Shin-yo Muto, and Masanobu Abe. A location predictor based on dependencies between multiple lifelog data. In *Proceedings of the 2010 International Workshop on Location Based Social Networks, LBSN 2010, November 2, 2010, San Jose, CA, USA, Proceedings*, pages 11–17, 2010.

[128] E. Ohn-Bar and M. Trivedi. Looking at humans in the age of self-driving and highly automated vehicles. *IEEE Trans. Intell. Veh.*, 1(1), 2016.

[129] Faisal Orakzai, Toon Calders, and Torben Bach Pedersen. Distributed convoy pattern mining. In *IEEE MDM*, pages 122–131, 2016.

[130] Faisal Orakzai, Toon Calders, and Torben Bach Pedersen. k/2-hop: Fast mining of convoy patterns with effective pruning. *PVLDB*, 12(9):948–960, 2019.

[131] J. K. Ord and Arthur Getis. Local spatial autocorrelation statistics: Distributional issues and an application. *Geographical Analysis*, 27(4):286–306, October 1995.

[132] J. Paefgen, F. Kehr, Y. Zhai, and F. Michahelles. Driving behavior analysis with smartphones: Insights from a controlled field study. In *Proc. 11th Int. Conf. Mobile and Ubiquitous Multimedia (MUM)*, 2012.

[133] Costas Panagiotakis, Eleni Kokinou, and Filippos Vallianatos. Automatic p-phase picking based on local-maxima distribution. *IEEE Trans. Geoscience and Remote Sensing*, 46(8):2280–2287, 2008.

[134] Costas Panagiotakis, Nikos Pelekis, Ioannis Kopanakis, Emmanuel Ramasso, and Yannis Theodoridis. Segmentation and sampling of moving object trajectories based on representativeness. *IEEE Transactions on Knowledge and Data Engineering TKDE*, 24(7):1328–1343, 2012.

[135] Costas Panagiotakis and Georgios Tziritas. A speech/music discriminator based on RMS and zero-crossings. *IEEE Trans. Multimedia*, 7(1):155–166, 2005.

[136] Luca Pappalardo et al. Returners and explorers dichotomy in human mobility. *Nature communications*, 6:8166, 2015.

[137] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678, June 2018.

[138] SeongHyeon Park, Byeongdo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, pages 1672–1678, 2018.

[139] Adam Pavlıček et al. A compact view of isochores in the draft human genome sequence. *FEBS letters*, 511(1-3):165–169, 2002.

[140] Philip Pecher, Michael Hunter, and Richard Fujimoto. Data-driven vehicle trajectory prediction. In *Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '16, pages 13–22, New York, NY, USA, 2016. ACM.

[141] Nikos Pelekis, Ioannis Kopanakis, Evangelos E. Kotsifakos, Elias Frentzos, and Yannis Theodoridis. Clustering uncertain trajectories. *Knowl. Inf. Syst.*, 28(1):117–147, 2011.

[142] Nikos Pelekis, Ioannis Kopanakis, Costas Panagiotakis, and Yannis Theodoridis. Unsupervised trajectory sampling. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD, Part III*, pages 17–33, 2010.

[143] Nikos Pelekis, Panagiotis Tampakis, Marios Vodas, Christos Doulkeridis, and Yannis Theodoridis. On temporal-constrained sub-trajectory cluster analysis. *Data Min. Knowl. Discov.*, 31(5):1294–1330, 2017.

[144] Nikos Pelekis, Panagiotis Tampakis, Marios Vodas, Costas Panagiotakis, and Yannis Theodoridis. In-dbms sampling-based sub-trajectory clustering. In *EDBT*, pages 632–643, 2017.

[145] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. *Machine Learning, p*, 01 2002.

[146] Petros Petrou, Panagiotis Nikitopoulos, Panagiotis Tampakis, Apostolos Glenis, Nikolaos Koutroumanis, Georgios M. Santipantakis, Kostas Patroumpas, Akrivi Vlachou, Harris V. Georgiou, Eva Chondrodima, Christos Doulkeridis, Nikos Pelekis, Gennady L. Andrienko, Fabian Patterson, Georg Fuchs, Yannis Theodoridis, and George A. Vouros. ARGO: A big data framework for online trajectory prediction. In *Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, August 19-21, 2019*, pages 194–197, 2019.

[147] Petros Petrou, Panagiotis Tampakis, Harris V. Georgiou, Nikos Pelekis, and Yannis Theodoridis. Online long-term trajectory prediction based on mined route patterns. In *MASTER@ECML-PKDD 2019*, pages 34–49, 2019.

[148] B. Porat. *Digital processing of random signals: Theory and methods*. Prentice Hall, Englewood Cliffs, NJ, USA, 1994.

[149] J. Przybyla, J. Taylor, J. Jupe, and X. Zhou. Estimating risk effects of driving distraction: A dynamic errorable car-following model. *Transp. Res. C*, 50, 2015.

[150] H. Qin, Y. Peng, and W. Zhang. Vehicles on rfid: Error-cognitive vehicle localization in gps-less environments. *IEEE Transactions on Vehicular Technology*, 66(11):9943–9957, Nov 2017.

[151] Disheng Qiu, Paolo Papotti, and Lorenzo Blanco. Future locations prediction with uncertain data. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I*, pages 417–432, 2013.

[152] M Reza Rahimi, Nalini Venkatasubramanian, and Athanasios V Vasilakos. Music: Mobility-aware optimal service allocation in mobile cloud computing. In *2013 IEEE sixth international conference on cloud computing*, pages 75–82. IEEE, 2013.

[153] Vasily E Ramensky et al. Dna segmentation through the bayesian approach. *Journal of Computational Biology*, 7(1-2):215–231, 2000.

[154] Charles S ReVelle and Ralph W Swain. Central facilities location. *Geographical analysis*, 2(1):30–42, 1970.

[155] Salvatore Rinzivillo et al. The purpose of motion: Learning activities from individual mobility networks. In *DSAA*, pages 312–318. IEEE, 2014.

[156] W. Rongben, G. Lie, T. Bingliang, and J. Lisheng. Monitoring mouth movement for driver fatigue or distraction with one camera. In *Proc. 7th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2004.

[157] A. Rossi, G. Barlacchi, M. Bianchini, and B. Lepri. Modelling taxi drivers' behaviour for the next destination prediction. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2019.

[158] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[159] Dimitris Sacharidis, Kostas Patroumpas, Manolis Terrovitis, Verena Kantere, Michalis Potamias, Kyriakos Mouratidis, and Timos K. Sellis. On-line discovery of hot motion paths. In *Proceedings of the 11th International Conference on Extending Database Technology, EDBT*, pages 392–403, 2008.

[160] F. Sagberg, G.F. Bianchi, and J. Engstrom. A review of research on driving styles and road safety. *Hum. Factors*, 57(7), 2015.

[161] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T. Campbell. Nextplace: A spatio-temporal prediction framework for pervasive systems. In *Pervasive Computing - 9th International Conference, Pervasive 2011, San Francisco, CA, USA, June 12-15, 2011. Proceedings*, pages 152–169, 2011.

[162] Kazuhiro Seki, Ryota Jinno, and Kuniaki Uehara. Parallel distributed trajectory pattern mining using hierarchical grid with mapreduce. *IJGHPC*, 5(4):79–96, 2013.

[163] R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 01 1973.

[164] Katarzyna Siła-Nowicka et al. Analysis of human mobility patterns from gps trajectories and contextual information. *IJGIS*, 30(5):881–906, 2016.

[165] M.R. Spiegel, J. Liu, and S. Lipschutz. *Mathematical Handbook of Formulas and Tables (4th/Ed.)*. McGraw-Hill, 2012.

[166] M.R. Spiegel, J. Schiller, and R.A. Srinivasan. *Probability and Statistics (3rd/Ed.)*. McGraw-Hill, 2009.

[167] Z. Sun and X.J. Ban. Vehicle classification using gps data. *Transp. Res. C*, 37, 2013.

[168] P. Tampakis, N. Pelekis, C. Doulkeridis, and Y. Theodoridis. Scalable distributed subtrajectory clustering. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 950–959, 2019.

[169] Panagiotis Tampakis, Christos Doulkeridis, Nikos Pelekis, and Yannis Theodoridis. Distributed subtrajectory join on massive datasets. *ACM Trans. Spatial Algorithms Syst.*, 6(2), 2020.

[170] Panagiotis Tampakis, Christos Doulkeridis, Nikos Pelekis, and Yannis Theodoridis. Distributed subtrajectory join on massive datasets. *ACM Trans. Spatial Algorithms and Systems*, To appear.

[171] Panagiotis Tampakis, Nikos Pelekis, Natalia V. Andrienko, Gennady L. Andrienko, Georg Fuchs, and Yannis Theodoridis. Time-aware sub-trajectory clustering in hermes@postgresql. In *ICDE*, pages 1581–1584, 2018.

[172] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2018.

[173] Lu An Tang, Yu Zheng, Jing Yuan, Jiawei Han, Alice Leung, Chih-Chieh Hung, and Wen-Chih Peng. On discovery of traveling companions from streaming trajectories. In *ICDE*, pages 186–197, 2012.

[174] MingJie Tang, Yongyang Yu, Qutaibah M. Malluhi, Mourad Ouzzani, and Walid G. Aref. Locationspark: A distributed in-memory data management system for big spatial data. *PVLDB*, 9(13):1565–1568, 2016.

[175] X. Tang. Texture information in run-length matrices. *IEEE Trans. Im. Proc.*, 7(11), 1998.

[176] Fernando Terroso-Saenz, Mercedes Valdes-Vela, and Antonio F Skarmeta-Gomez. Online route prediction based on clustering of meaningful velocity-change areas. *Data mining and knowledge discovery*, 30(6):1480–1519, 2016.

[177] Evimaria Terzi and Panayiotis Tsaparas. Efficient algorithms for sequence segmentation. In *SDM*, pages 316–327. SIAM, 2006.

[178] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 4th edition, 2008.

[179] Roberto Trasarti et al. Mining mobility user profiles for car pooling. In *KDD*, pages 1190–1198. ACM, 2011.

[180] Roberto Trasarti et al. Myway: Location prediction via mobility profiling. *IS*, 64:350–367, 2017.

[181] Roberto Trasarti, Riccardo Guidotti, Anna Monreale, and Fosca Giannotti. Myway: Location prediction via mobility profiling. *Inf. Syst.*, 64:350–367, 2017.

[182] Md Reaz Uddin, Chinya Ravishankar, and Vassilis J Tsotras. Finding regions of interest from trajectory data. In *Proceedings of the 12th International Conference on Mobile Data Management MDM*, volume 1, pages 39–48, 2011.

[183] Angelos Valsamis, Konstantinos Tserpes, Dimitrios Zissis, Dimosthenis Anagnostopoulos, and Theodora A. Varvarigou. Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction. *J. Syst. Softw.*, 127:249–257, 2017.

[184] Juan Van Roy, Niels Leemput, Sven De Breucker, Frederik Geth, Peter Tant, and Johan Driesen. An availability analysis and energy consumption model for a flemish fleet of electric vehicles. In *EEVC European Electric Vehicle Congress*, 2011.

[185] Marcos R. Vieira, Petko Bakalov, and Vassilis J. Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *ACM SIGSPATIAL*, pages 286–295, 2009.

[186] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684, 2002.

[187] C. Wang, L. Ma, R. Li, T. S. Durrani, and H. Zhang. Exploring trajectory prediction through machine learning methods. *IEEE Access*, 7:101441–101452, 2019.

[188] W. Wang, J. Xi, A. Chong, and L. Li. Driving style classification using a semisupervised support vector machine. *IEEE Trans. Human–Mach. Syst.*, 47(5), 2017.

[189] W. Wang, J. Xi, and D. Zhao. Driving style analysis using primitive driving patterns with bayesian nonparametric approaches. *IEEE Trans. on Intell. Trans. Sys.*, 20(8), 2019.

[190] Y. Wang, D. Zhang, Y. Liu, and K. Tan. Trajectory forecasting with neural networks: An empirical evaluation and a new hybrid model. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2019.

[191] Josh Warren, Jeff Lipkowitz, and Vadim Sokolov. Clusters of driving behaviour from observational smartphone data. *IEEE Intell. Trans. Sys. Mag.*, 11(3), 2019.

[192] Randall T. Whitman, Michael B. Park, Bryan G. Marsh, and Erik G. Hoel. Spatio-temporal join on apache spark. In *Proceedings of the 25th ACM International Conference on Advances in Geographic Information Systems, SIGSPATIAL*, pages 20:1–20:10, 2017.

[193] Fan Wu, Kun Fu, Yang Wang, Zhibin Xiao, and Xingyu Fu. A spatial-temporal-semantic neural network algorithm for location prediction on moving objects. *Algorithms*, 10:37, 03 2017.

[194] Yongyi Xian, Yan Liu, and Chuanfei Xu. Parallel gathering discovery over big trajectory data. In *Proceedings of the 2016 IEEE International Conference on Big Data, BigData*, pages 783–792, 2016.

[195] Dong Xie, Feifei Li, and Jeff M. Phillips. Distributed trajectory similarity search. *PVLDB*, 10(11):1478–1489, 2017.

[196] Zhixian Yan et al. Semantic trajectories: Mobility data computation and annotation. *ACM TIST*, 4(3):49, 2013.

[197] Gökhan Yavas, Dimitrios Katsaros, Özgür Ulusoy, and Yannis Manolopoulos. A data mining approach for location prediction in mobile environments. *Data Knowl. Eng.*, 54(2):121–146, 2005.

[198] W. Zhan, A. L. de Fortelle, Y. Chen, C. Chan, and M. Tomizuka. Probabilistic prediction from planning perspective: Problem formulation, representation simplification and evaluation metric. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1150–1156, June 2018.

[199] T. Zhang, R. Raghu, and L. Miron. Birch: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data. Canada (SIGMOD)*, 1996.

[200] PX Zhao, K Qin, Q Zhou, CK Liu, and YX Chen. Detecting hotspots from taxi trajectory data using spatial cluster analysis. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(4):131–135, 2015.

[201] Kai Zheng, Yu Zheng, Nicholas Jing Yuan, and Shuo Shang. On discovery of gathering patterns from trajectories. In *ICDE*, pages 242–253, 2013.

[202] Yu Zheng. Trajectory data mining: An overview. *ACM TIST*, 6(3):29:1–29:41, 2015.

[203] Yu Zheng et al. Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, 5(1):5, 2011.

[204] Nikolaos Zorbas, Dimitrios Zissis, Konstantinos Tserpes, and Dimosthenis Anagnostopoulos. Predicting object trajectories from high-speed streaming data. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 2*, pages 229–234, 2015.

[205] Nikolaos Zygouras and Dimitrios Gunopulos. Discovering corridors from GPS trajectories. In *ACM SIGSPATIAL*, pages 61:1–61:4, 2017.

[206] Nikolaos Zygouras and Dimitrios Gunopulos. Corridor learning using individual trajectories. In *IEEE MDM*, pages 155–160, 2018.